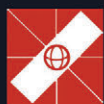


High Performance Networking VII

Edited by
Ahmed N. Tantawy



IFIP



SPRINGER-SCIENCE+BUSINESS MEDIA, B.V.

High Performance Networking VII

Visit our IT & Applied Computing resource centre
www.IT-CH-com

IFIP – The International Federation for Information Processing

IFIP was founded in 1960 under the auspices of UNESCO, following the First World Computer Congress held in Paris the previous year. An umbrella organization for societies working in information processing, IFIP's aim is two-fold: to support information processing within its member countries and to encourage technology transfer to developing nations. As its mission statement clearly states,

IFIP's mission is to be the leading, truly international, apolitical organization which encourages and assists in the development, exploitation and application of information technology for the benefit of all people.

IFIP is a non-profitmaking organization, run almost solely by 2500 volunteers. It operates through a number of technical committees, which organize events and publications. IFIP's events range from an international congress to local seminars, but the most important are:

- the IFIP World Computer Congress, held every second year;
- open conferences;
- working conferences.

The flagship event is the IFIP World Computer Congress, at which both invited and contributed papers are presented. Contributed papers are rigorously refereed and the rejection rate is high.

As with the Congress, participation in the open conferences is open to all and papers may be invited or submitted. Again, submitted papers are stringently refereed.

The working conferences are structured differently. They are usually run by a working group and attendance is small and by invitation only. Their purpose is to create an atmosphere conducive to innovation and development. Refereeing is less rigorous and papers are subjected to extensive group discussion.

Publications arising from IFIP events vary. The papers presented at the IFIP World Computer Congress and at open conferences are published as conference proceedings, while the results of the working conferences are often published as collections of selected and edited papers.

Any national society whose primary activity is in information may apply to become a full member of IFIP, although full membership is restricted to one society per country. Full members are entitled to vote at the annual General Assembly, National societies preferring a less committed involvement may apply for associate or corresponding membership. Associate members enjoy the same benefits as full members, but without voting rights. Corresponding members are not represented in IFIP bodies. Affiliated membership is open to non-national societies, and individual and honorary membership schemes are also offered.

High Performance Networking VII

**IFIP TC6 Seventh International Conference
on High Performance Networks (HPN '97),
28th April – 2nd May 1997,
White Plains, New York, USA**

Edited by

Ahmed Tantawy

*IBM T.J. Watson Research Center,
Yorktown Heights, NY, USA*



SPRINGER-SCIENCE+BUSINESS MEDIA, B.V.

First edition 1997

© 1997 Springer Science+Business Media Dordrecht
Originally published by Chapman & Hall in 1997

ISBN 978-1-4757-5401-8 ISBN 978-0-387-35279-4 (eBook)
DOI 10.1007/978-0-387-35279-4

Apart from any fair dealing for the purposes of research or private study, or criticism or review, as permitted under the UK Copyright Designs and Patents Act, 1988, this publication may not be reproduced, stored, or transmitted, in any form or by any means, without the prior permission in writing of the publishers, or in the case of reprographic reproduction only in accordance with the terms of the licences issued by the Copyright Licensing Agency in the UK, or in accordance with the terms of licences issued by the appropriate Reproduction Rights Organization outside the UK. Enquiries concerning reproduction outside the terms stated here should be sent to the publishers at the London address printed on this page.

The publisher makes no representation, express or implied, with regard to the accuracy of the information contained in this book and cannot accept any legal responsibility or liability for any errors or omissions that may be made.

A catalogue record for this book is available from the British Library



CONTENTS

Preface	vii
Committees	ix
 PART ONE Multicast Implementation Issues	
1 Multicast server architectures for supporting IP multicast over ATM <i>R.R. Talpade and M.H. Ammar</i>	3
2 Center placement algorithms for large multicast groups <i>A. Weigmann, J. Nonnenmacher and E. Biersack</i>	18
 PART TWO Experimental Results	
3 On the integration of the UMTS and B-ISDN system <i>G. Karagiannis, B.J. van Beijnum and I.G.M.M. Niemegeers</i>	39
4 Measuring the behavior of a world-wide web server <i>J. Almeida, V. Almeida and D. Yates</i>	57
5 A framework for ATM testing <i>J. Micheel, C. Tittel and J. Tiemann</i>	73
 PART THREE Multimedia Traffic	
6 The effect of various ATM switch architectures on VBR video performance <i>R.P. Tsang, J. Hsieh and D.H.C. Du</i>	87
7 Throughout optimization for multimedia applications over high speed networks <i>S. Zeadally, G. Gheorghiu and A.F.J. Levi</i>	101
8 Issues in platform-independent support for multimedia desktop <i>O. Kim, P. Kabore, J.P. Favreau and H. Abdel-Wahab</i>	115
9 Design and implementation of a flexible traffic controller for ATM connections <i>L. Lamti, H. Afifi and M. Hamdi</i>	130
 PART FOUR Quality of Service	
10 On routing with QoS constraints in ATM networks <i>D. Cavendish and M. Gerla</i>	149
11 Mobiware: QoS-aware middleware for mobile multimedia communications <i>A. Campbell</i>	166

PART FIVE Fundamental Concepts

- | | | |
|----|---|-----|
| 12 | Enforcing quality of service for adaptive multimedia applications via fair queueing
<i>F. Toutain</i> | 187 |
| 13 | Adaptive variation of reliability
<i>R. Kravets, K. Calvert, P. Krishnan and K. Schwan</i> | 202 |
| 14 | On compartmental modelling of multi-service communication networks
<i>M. Sivabalan, H.T. Mouftah and G. Takahara</i> | 217 |
| 15 | Performance evaluations of partial order connections
<i>M. Fournier, C. Chassot, M. Diaz and A. Lozes</i> | 232 |

PART SIX Architectural Issues

- | | | |
|----|--|-----|
| 16 | A performance model for integrated layer processing
<i>B. Ahlgren</i> | 249 |
| 17 | An architecture for active networking
<i>S. Bhattacharjee, K. Calvert and E. Zegura</i> | 265 |
| 18 | The strategy of traffic dispersion
<i>E. Gustafsson and G. Karlsson</i> | 280 |

PART SEVEN Bandwidth Allocation

- | | | |
|----|--|-----|
| 19 | Dynamic bandwidth allocation for stored VBR video in ATM endsystems
<i>S. Gumbrich, H. Emgrunt and T. Braun</i> | 297 |
| 20 | Analysis of a new end-to-end proportional bandwidth allocation algorithm
<i>L. Huynh and A. Nilsson</i> | 318 |
| 21 | A simulation study of a wireless bandwidth reservation multiple access protocol for multimedia traffic
<i>Z. Zhang, I. Habib and T. Saadawi</i> | 337 |

Index of contributors	353
------------------------------	-----

Keyword index	354
----------------------	-----

Preface

It is always confusing, and perhaps inconvenient at times, using generic terms that will mean something to everyone but different things to different people. "High Performance" is one of those terms. High Performance can be viewed as synonymous to High Speed or Low Latency or a number of other characteristics. The interesting thing is that such ambiguity can sometimes be useful in a world where focus shifts quite easily from one issue to another as times and needs evolve.

Many things have changed since the first HPN conference held in Aachen, Germany in 1987. The focus then was mainly on Media Access Control (MAC) protocols that allow users to share the high bandwidth of optical fiber. FDDI (Fiber Distributed Data Interface) was making its debut with its amazing 100 Mbps speed. ATM (Asynchronous Transfer Mode) and SONET (the Synchronous Optical Network) were beginning to capture our imagination. What could users possibly do with such "high performance"? Share it!

After realizing that the real problems had gradually shifted away from the network media to the periphery of the network, focus also began to shift. Adapter design, protocol implementation, and communication systems architecture began to attract our interest. Networking -not Networks- became the hot issue.

The world around us has also changed. The technical community kept the power of networking quietly devoted to its own use. We developed that technology over the past three decades and we are perfecting it. We used it to communicate and transfer information among ourselves. Then, suddenly, the word got out and people started to wonder. What a great world this would be if we can all communicate electronically! They saw us and they liked what they saw us doing (and we did some selling too). Today, the word Internet is a household name. Elementary school children send email and exchange pictures and videos to their e-pen pals across the globe. Older kids rely on the web to get the information they need to write their papers.

New challenges appeared. High Performance Networking is not about sharing the local fiber anymore. It is also much less about the architecture of end systems. It is now more about the application, i.e., the use of the high bandwidth network infrastructure. Issues like quality of service, mobility, multimedia communication, security, scalability, multicasting and large group collaboration are attracting a lot of our attention. But, naturally, some fundamental and architectural concepts are still being developed to help us build better networking systems.

The evolution is continuing and it will be interesting to see what the next ten years of HPN research will bring to the technical community and to the society at large. Peace and prosperity is what we all hope for but almost never write about in our papers. We are fortunate to be at the core of the most influential

technology of this decade. Let's make the world a better and safer place through communication.

HPN '97 is the seventh in a very successful conference series and the first to be held outside Europe. The program includes 21 carefully selected papers and 11 invited presentations. This is a new formula for a new and busy age. Due to the increasingly experimental nature of HPN research today and the great diversity of ideas, approaches and trends in this field, we devoted one third of the conference to presentations given by known leaders in the field who have accepted to share their most recent experiences and views with the attendees in a less formal atmosphere. Formal papers remain however the core of the program and are the only ones printed in this book for a wide dissemination of the more refined and proven ideas that they present.

It is a true honor to serve as the General Chair of HPN '97. It is also a great privilege to work with the distinguished members of the international Program Committee who helped me refine the philosophy of this event and volunteered their precious time to review the submitted papers, invite speakers, and design the program. The Local Organization Committee is putting a tremendous amount of dedicated effort into developing an exciting environment that facilitates the exchange of ideas, the development of new connections, and the retrieval of some lost links. I know that the dedication of these outstanding volunteers is highly appreciated by all of us and shows that we, in the HPN arena, are truly high performance people!

If you are an attendee of HPN '97, welcome and enjoy the company! If you are reading this book for the great work it contains and the innovative ideas it presents, I hope you will find it very useful and informative.
Good luck to all.

Ahmed N. Tantawy
Yorktown Heights,
April 1997

HPN '97 Committees

Conference Chairman:

Ahmed Tantawy (IBM - T.J. Watson, USA)

PROGRAM COMMITTEE MEMBERS:

Ian Akyildiz (Georgia Tech, USA)
Mostafa Ammar (Georgia Tech, USA)
Harmen van As (Tech. U. Vienna, Austria)
Pat Baker (Hewlett-Packard Labs, UK)
Jean-Yves le Boudec (EPF Lausanne, Switzerland)
Augusto Casaca (INESC, Portugal)
Andre Danthine (U. Liege, Belgium)
Bruce Davie (Cisco Systems, USA)
Gary Delp (IBM, USA)
Michel Diaz (LAAS-CNRS, France)
Christophe Diot (INRIA, France)
Otto Duarte (U. Fed. Rio de Janeiro, Brazil)
Serge Fdida (U. Paris VI, France)
Zygmunt Haas (Cornell U., USA)
Salim Hariri (Syracuse U., USA)
Marjory Johnson (NASA-RIACS, USA)
Dae Young Kim (Chung Nam U., Korea)
Kurt Maly (Old Dominion U., USA)
Hussein Mouftah (Queens U., Canada)
Guru Parulkar (Washington U. St. Louis, USA)
Stephen Pink (SICS, Sweden)
Radu Popescu-Zeletin (GMD Fokus, Germany)
Ramon Puigjanier (U. Illes Balears, Spain)
Guy Pujolle (U. Versailles, France)
KK Ramakrishnan (AT&T Research, USA)
Doug Shepherd (U. Lancaster, UK)
Jonathan Smith (U. Pennsylvania, USA)
Otto Spaniol (U. Aachen, Germany)
James Sterbenz (GTE, USA)
Fouad Tobagi (Stanford U., USA)
Giorgio Ventre (U. Napoli, Italy)
Shukri Wakid (NIST, USA)
Raj Yavatkar (Intel, USA)

Yechiam Yemini (Columbia U., USA)

Martina Zitterbart (U. Braunschweig, Germany)

LOCAL ORGANIZATION COMMITTEE:

Jurij Paraszczak (IBM - T.J. Watson, USA), Chairman

Andrew Campbell (Columbia U., USA)

Charles Castellano (Prodigy, USA)

Asser Tantawi (IBM - T.J. Watson, USA)

Marc Willebeek-LeMair (IBM - T.J. Watson, USA)

PART ONE

Multicast Implementation Issues

Multicast Server Architectures for Supporting IP Multicast over ATM

Rajesh R. Talpade and Mostafa H. Ammar

*Networking and Telecommunications Group, College of Computing
Georgia Institute of Technology, Atlanta, GA 30332.*

email: {taddy, ammar}@cc.gatech.edu

Abstract

The Multicast Address Resolution Server architecture has been proposed as a mechanism for supporting IP multicast over ATM networks. Two basic techniques exist within this architecture for the intra-subnet multicasting of IP packets over ATM networks. One approach makes use of a mesh of point-to-multipoint Virtual Circuits (*VC Mesh*) each of which is rooted at a multicast source, while the other uses a shared point-to-multipoint tree rooted at a Multicast Server (*MCS*). In this paper we describe the design and implementation of a MARS client and an MCS. We present a framework for comparing the VC Mesh and MCS approaches using experimental and simulation techniques. Finally we present three protocols for the usage of multiple MCSs per IP multicast group, a technique required for groups with large number of senders and group members. * *

Keywords

IP Multicast, MCS Architectures, Network Protocols, MBONE, ATM

1 INTRODUCTION

Asynchronous Transfer Mode (ATM) is a connection-oriented non-broadcast multiple access (NBMA) technology which is increasingly being used for local and wide area networking. A Virtual Circuit (VC) has to be established between participating hosts on an ATM network before transfer of data can take place. Unicast communication is over a point-to-point (unidirectional or bidirectional) VC which has been setup between the two communicating peers, whereas multicast communication is over a point-to-multipoint *unidirectional* VC which originates at the sender and has the group members as its leaves. The sender has to explicitly add each of the group members to this VC as the ATM Forum's currently published signaling specifications (UNI 3.1, (1995))* do not provide a multicast address abstraction.

The absence of an ATM multicast address presents a problem for mapping a

*This work was supported in part by Bellcore.

*Extended version at <http://www.cc.gatech.edu/computing/Telecomm/playground/MCS>

*The latest ATM specification as defined in UNI 4.0 also suffers from a variation of this problem. We discuss this in the extended version due to space limitations here.

layer 3 multicast address to the corresponding link-layer multicast address, as is possible when Ethernet is used as the link-layer. Thus a mechanism is needed for resolving layer 3 (e.g., IP) multicast addresses into the corresponding set of ATM addresses of multicast group members, before a point-to-multipoint unidirectional Virtual Circuit (VC) can be set up for multicasting IP data. The Internetworking over NBMA (ION) working group of the Internet Engineering Task Force (IETF) has proposed the Multicast Address Resolution Server (MARS) protocol as a solution to this problem (1996a). The MARS server is used to maintain the mapping between IP multicast group addresses and the corresponding ATM addresses. Hosts in the ATM network use the MARS to resolve IP multicast addresses into corresponding lists of group members. The source has two options for multicasting data to the group members. It can set up a point-to-multipoint VC from itself to each of the group members, and then proceed to send data out on it (the *VC mesh* approach). Or it can make use of a proxy Multicast Server (MCS). The source transmits data to such an MCS, which in turn uses a point-to-multipoint VC to get the data to the group members (the *MCS* approach).

The VC mesh approach, along with the MARS server and client architectures, have been described in detail in (1996a). It outlines the broad requirements for a simple MCS, but does not specify the architecture details. This paper presents an overview of the MCS architecture, and explores the implications of using the MCS approach. To that end we first report on the implementation of a MARS client that was needed to provide a complete implementation of the MARS architecture. We then provide an experimental and simulation-based framework for comparing the VC Mesh and MCS approaches. The simulation makes use of both actual MBONE and simulated group dynamics. Such a comparative framework is necessary to ease the choice of using either approach on a per group basis. The MCS approach is shown to be better than the VC Mesh from the VC consumption and signaling load perspective. This suggests that the MCS approach should be used for multicasting IP data in ATM networks, especially for large IP multicast groups. However, as we experimentally show, the MCS can be a bottleneck for data traffic under certain conditions. This problem, and the associated problem of the central MCS being a single point of failure, can be solved by using multiple MCSs per IP multicast group. Towards this end, we finally propose three protocols with varying capabilities and requirements for the support of multiple MCSs.

The paper is organized as follows. Section 2 presents an overview of the MARS architecture, along with an overview of our MARS client implementation design. A simulation-based comparison of the VC Mesh and MCS approaches is presented in section 3, and performance experiments are presented in section 4. We discuss the three multiple MCS mechanisms in section 5, and summarize our work in section 6.

2 THE MARS ARCHITECTURE

The MARS architecture is based on a client/server model. It manages a *cluster* of ATM-attached endpoints, with a cluster being defined as the set of ATM interfaces that choose to use the same MARS to register their memberships and receive their updates. The ATM interface of each host in the cluster is assigned a unique IP address, with the hosts thus forming a Logical IP Subnet (LIS). The distribution of multicast group membership information between MARS and the clients is achieved by MARS messages.

2.1 Summary of the VC Mesh approach

When an application at a client in the cluster joins/leaves a group, the client has to register this event with the MARS. The MARS thus acts as a registry, maintaining a mapping of IP multicast group addresses and corresponding ATM addresses. Each of these mappings indicates the ATM addresses of clients which have joined that particular group. A source in the LIS which needs to transmit data to group members first sends a request to the MARS. The MARS responds with the ATM addresses of the clients that are members of the particular group. The source opens a point-to-multipoint VC to the clients and transmits data on this outgoing VC. The source can add/delete clients from the outgoing VC as it receives updates from the MARS whenever a client joins/leaves the group. The outgoing VC from the source is closed on absence of traffic for the group.

2.2 Summary of MCS Architecture and Operation

The MCS acts as a proxy server, retransmitting data received from senders to the group members. The simplest MCS architecture involves taking incoming AAL_SDUs from the multicast sources and sending them out over a point-to-multipoint VC to the group members. A more efficient design would require the MCS to look inside the received AAL_SDUs and determine which IP multicast group they are destined for. A single instance of such an MCS could support several multicast groups, and have one outgoing point-to-multipoint VC for each of the groups.

The MCS opens a point-to-point bidirectional VC with the MARS (the identity of which must be known) on startup. This VC is used by the MARS for sending all control messages specific to this MCS, and by the MCS for sending control messages to the MARS. The MCS registers itself with the MARS on startup. The MARS then adds the MCS to a point-to-multipoint unidirectional VC (Server Control VC) that it maintains to each MCS in the cluster. This VC is used by the MARS to disseminate general cluster information to all the MCSs.

After registering itself, an MCS can register to support group(s) with the MARS. On successful registration to support a group, it gets the ATM ad-

dresses of current group members from the MARS, and opens a point-to-multipoint VC to them. The MCS thus has a separate outgoing point-to-multipoint VC for each group (with non-zero membership) that it has registered to support.

Senders to the group transmit data to the MCS, which then retransmits the received data on the outgoing point-to-multipoint VC. If the MCS is supporting multiple groups simultaneously, it looks inside the received AALSDUs (from the senders) before retransmitting it on the appropriate outgoing VC. The MARS transmits group membership change information to the MCSs on the Server Control VC, which enables them to keep their outgoing VCs updated.

2.3 The Implementation Design

We implemented the client-end of the architecture and the MCS to support IP Multicast, which can be best thought of as a “shim”, layer sitting between IP’s link layer interface and the underlying UNI 3.1 service. *Ipmcatmd* has to establish a point-to-point bidirectional VC, which it uses to send queries to and receive replies from the MARS. It also terminates the point-to-multipoint VC (Cluster Control VC), originating from the MARS and terminated by all clients, which is used to disseminate group change information (Design described in extended version).

Ipmcatmd has been implemented as a user-level daemon which executes at each of the hosts. Bellcore’s experimental UNI 3.0/3.1 signaling stack (Q.PortTM Portable ATM Signaling Software) is used as the signaling entity. *Ipmcatmd* communicates with Q.PortTM software, which also executes at user-level, through a TCP socket. In addition to the user-level code, *ipmcatmd* has an associated streams module. This is used by the IP link layer interface to communicate with *ipmcatmd*.

The MCS was implemented entirely in user space under SunOS4.1.3. It conformed to the specifications in (1997). Bellcore provided the user-space MARS server code (1996c) used in our experiments. The server implementation conformed to the specifications in (1996a). As with the MARS client, Q.PortTM software was used to provide host side UNI signaling for both the MARS and the MCS.

3 A FRAMEWORK FOR COMPARING THE VC MESH AND MCS APPROACHES

In the existing MARS architecture, the decision to use the VC Mesh or the MCS approach can be done in an LIS on a per group basis. This choice for each group can be made optimally if a comprehensive framework exists for comparing the two approaches. We present such a framework in the following discussion.

3.1 VC Consumption

VCs can be an expensive and scarce resource in an ATM network. Users may be billed on a per VC basis, in which case it is economically desirable to maintain a low VC count. VCs consume memory and other resources at end hosts, which places practical limits on the maximum number of VCs that can be used at an ATM interface. This limit is on the order of 1000 VCs for current generation ATM Network Interface Cards (NICs). Each additional VC also increases the signaling load that the switches have to support. Considering these reasons, it is important to design and use applications such that VC consumption is minimized, both at the end-hosts (because of ATM NIC limitations) and in the ATM network (to reduce the signaling load and resource usage at switches).

We first attempt to understand the implications of using the MARS architecture in an increasingly ATM-based Internet from the VC consumption perspective. We use the group dynamics of the current MBONE as measured using *mlisten* (1996) and compute the VC consumption when the MBONE hosts are connected to an ATM-based Internet. We then attempt to compute the VC consumption for future Internet multicast applications using a simulation.

(a) VC consumption in the MBONE

The Multicast Backbone (MBONE) is the virtual network of multicast-capable routers and end-hosts in the Internet. A good approximation of VC consumption and the load on the MARS server resulting from running the MBONE over an ATM-based Internet (machines interconnected using ATM) can be obtained by using the current MBONE group dynamics. These MBONE group dynamics have been obtained using the “*mlisten*” tool. The statistics gathered by “*mlisten*” indicate the host address, the IP Multicast session that the host is a member of or sender to, the start time and the duration of activity (sending or receiving). These statistics are collected for each host in the Internet that is receiving or sending IP multicast data using the MBONE “*sdr*” application. The data used for our study was gathered from February 22nd through March 10th 1996, which includes an IETF meeting and a NASA shuttle session.

The trace data was used to drive a simulation where the MARS architecture was being used over the entire Internet. A count was maintained of the number of point-to-point and point-to-multipoint VCs that were being used. This included both the data VCs and the control VCs from the MARS. This analysis was done for both the VC Mesh and the MCS approach, with one MCS per group being assumed. The data included a total of 1719 distinct hosts, 9 distinct groups, and a maximum of 10 senders (1 or 2 per group). Figure 1 presents the group membership, the point-to-multipoint VC consumption (for both VC Mesh and MCS) and the variation in point-to-multipoint VCs (for both VC Mesh and MCS).

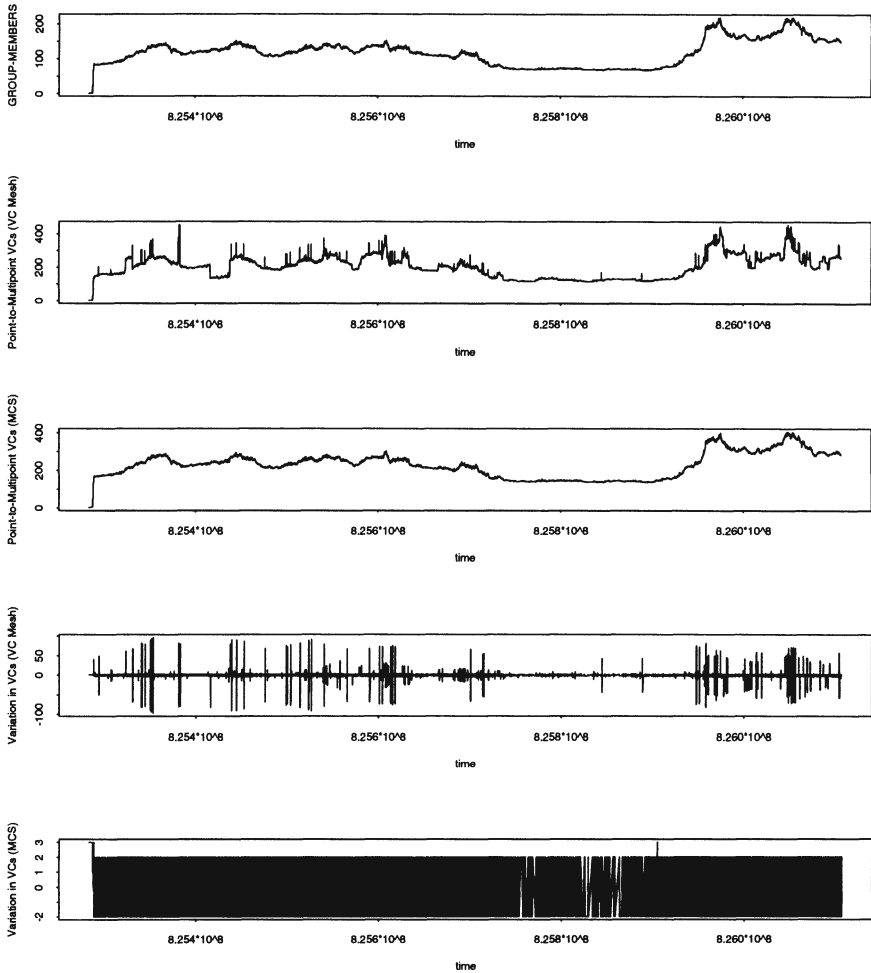


Figure 1 VC consumption in the MBONE.

The number of point to multipoint VCs are comparable for the VC Mesh and the MCS case. This is due to the low number of senders per group (1 or 2), which reduces the advantage of the MCS. Also we consider control VCs in our computation, and the number of hosts are comparable to the number of group members. Hence data VCs do not exert an unduly large influence on the VC count in this case. However the variation in the number of point to multipoint VCs is considerably larger in the VC Mesh than in the MCS case, even for such small group membership values. This is because, in the VC Mesh case, all group members are added(dropped) from a VC originating from a sender when that sender starts(stops) transmitting to the group. On the other hand, in the MCS case, only the MCS gets added(dropped) from this VC. This enforces the notion that the signaling load imposed by the VC

App	# of grps	Grp Sz	# of snds	Grp duration	Sync
1	1000	10	10	4000sec	No
2	1000	10	10	days	No
3	10	1000	1-10	12000sec	No
4	100	10	1	180sec	Yes
5	100	100	1	10800sec	Yes

Table 1 Application characterization

Mesh is significantly larger than by the MCS case, even if the number of VCs being used are not appreciably different.

(b) VC consumption for future Internet multicast applications

IP Multicast has not yet been widely used in the Internet, partly due to bandwidth restrictions and partly due to older software and hardware. However this experimental state is expected to end soon, and a need exists to determine the implications of using more sophisticated IP Multicast-based applications over the ATM-based Internet. We present an approach for characterizing such applications here, and determine the VC consumption and other parameters in the ATM-based Internet by means of a simulation.

Our approach uses group size, number of senders, number of groups, group duration and "synchronicity of sessions" (whether all senders/receivers start/end simultaneously) to characterize applications. Depending on these parameters, IP Multicast applications can be placed in five categories (1:Videoconference, 2:Database update, 3:Mbone-like, 4:Multicast FTP, 5:Seminar) as indicated in table 1. The simulation used these parameters to generate groups, group members and senders according to an inter-arrival rate computed using Little's Law (*average interarrival rate = average duration/average size*). The actual inter-arrival rate was thus uniformly distributed with a mean at the computed inter-arrival rate. The duration of senders(receivers) was fixed to be 25% of the group duration for applications 1 and 3, while it was equal to the group duration for 4 and 5. Application 2 represents ambient multicast activity, and was not considered in our computations. The number of end-hosts of the ATM network was assumed to be 100000 in the simulation, with each time unit corresponding to 1 second and the total length of the simulation being 1 day (86400 seconds). (We realize the practical limitations of a cluster size = 100000, and use it only as a representation of the estimated size of the Internet).

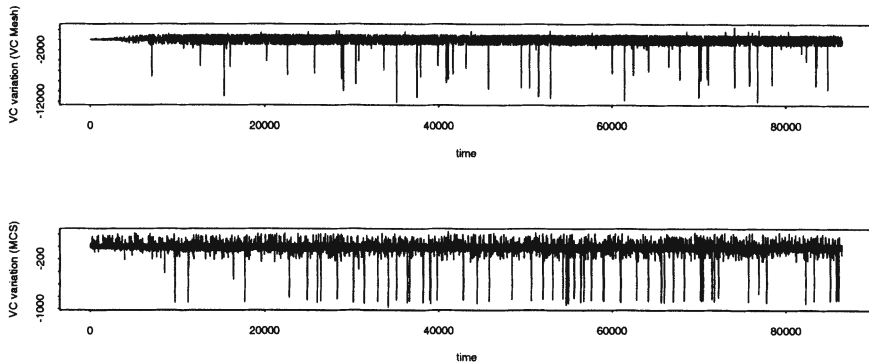


Figure 2 Variation in Point to Multipoint VCs.

Average values	VC Mesh	MCS
Number of Hosts	29945.936	31216.737
Number of Groups	1060.853	1086.119
Total Group members	23926.513	28643.441
Total senders	7513.724	7778.982
Total Point to Multipoint VCs	133985.948	45260.219
Point to Multipoint VCs per host	4.970	1.450

Table 2 Simulation Parameters

The simulation was used to compute the total number of point to multipoint VCs in the network and their variation per unit time, and the average number of VCs per active (sending/receiving IP Multicast data) host. The output parameters at steady state vary around an average value. This phenomenon corresponds to the flat part of the MBONE data graphs, where group dynamics are relatively stable. So we present average values of the output parameters from the simulation in table 2 instead of the graphs. We do present the variation in point to multipoint VCs (for VC Mesh and MCS) in graph form in figure 2. The first four parameters in table 2 indicate the actual values of the input parameters as were computed by the simulation, using input values from table 1. They are approximately and not exactly equal for the VC Mesh and the MCS case as they represent separate runs of the simulation.

The output data indicates that the total number of VCs in the VC Mesh case are about 30% more than in the MCS case. This result is to be contrasted with the one from the MBONE data, where the two were comparable as the number of senders were very low, which is not the case here. Also, the variation in VCs is considerably higher in the VC Mesh than the MCS case. The average number of VCs per active host remains low in either case, however the VC Mesh average is more than the MCS case.

3.2 Qualitative Comparison

The results of the simulation and the MBONE data analysis vindicate the fact that the VC Mesh imposes a significantly higher VC consumption and signaling load, both on the switches in the network and at individual end-hosts. A detailed qualitative comparison is presented in the extended version of this paper. We only outline the main points here due to space limitations.

The main advantages of using the MCS approach are summarized here. The solution of using IP multicast routers after fragmenting the cluster into several LISs (due to the VC Mesh's large VC consumption) has adverse performance implications, as will be discussed in section 4. Group membership changes cause a decreased level of signaling load to be generated at the switch in the MCS approach, as only the MCS has to add/delete a cluster member from its point-to-multipoint VC, instead of all the senders. The MCS approach provides a centralized control of multicast bandwidth usage over the ATM network, which is useful in cases where policy demands a limit on the share of bandwidth available for multicast purposes.

There are some disadvantages of using the MCS approach, however solutions do exist which minimize them. Data throughput and end-to-end latency may be adversely affected due to the additional level of indirection introduced by the MCS. The MCS can potentially become a bottleneck and a central point of failure. Both this problem and the previous one can be solved by using multiple MCSs per group. If a multicast source happens to be a member of the group it is transmitting to, it will receive a copy of the data back over the VC from the MCS. Thus additional header identification is needed for a source to discard such "bounced-back" data. This mechanism has been defined in (1996a) to be a 16 bit cluster member identifier (CMI). The need for each multicast source in the cluster to use differing quality of service (QOS) parameters for outgoing traffic cannot be satisfied by a simple MCS solution.

The preceding discussion demonstrates that the MCS approach is much better than the VC Mesh in terms of VC consumption, signaling load, number of groups possible in the LIS, and permissible size of the LIS. The "bounced-back data" problem and the QOS problem are solved by other proposals. The main problem with using MCSs is the possibility of a single MCS being a bottleneck for data, and a single point of failure. We propose a solution for both these problems using multiple MCSs in section 5.

4 MCS PERFORMANCE EXPERIMENTS

We conducted several experiments to compare the VC Mesh approach with the MCS approach. The testbed used for the experiments consisted Sparcstations (5 Sparc5s, 3 Sparc20s) running SunOS4.1.3, with Fore SBA200e NICs controlled by ForeThought version 3.0.3 (1.5). Two Fore ASX200 switches connected the machines through OC/3 UTP cable. The switches were inter-

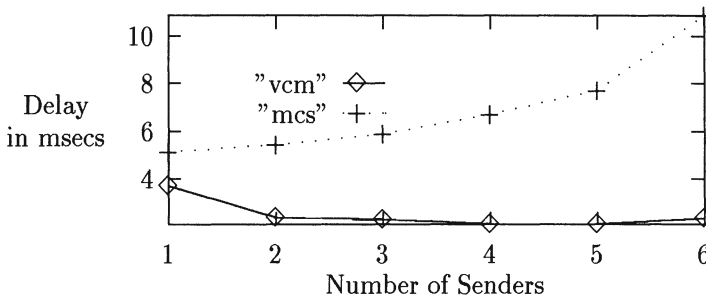


Figure 3 Impact of increasing load on VC Mesh and MCS approaches.

connected through OC/3 Multimode fiber. The Sparc5s were never used as receivers of data due to the known problems that they have with receiving ATM cells when running SunOS 4.1.3 and Fore software.

The first experiment compared the VC Mesh and MCS approaches as the number of senders is increased, with the goal of demonstrating the effect of increased load on the two approaches. We used one receiver (group member) in the LIS, with one through six senders transmitting simultaneously. The delay at the receiver to *receive* 20 MB of data was measured in all the cases.

As can be seen in figure 4, the delay in receiving the same number of bytes for the VC Mesh case actually decreases with increase in the number of senders. This is because the receiver gets an increasing amount of data with the increase in the number of senders, which causes it to receive 20 MB faster. This continues until it cannot receive data any faster, and then the effect of saturation of the incoming link to the receiver combined with the increase in interrupt frequency at the receiver causes the delay to actually increase, as is seen for six senders.

In the MCS case, both the delay and the rate of increase of the delay is directly proportional to the number of senders. This can be attributed to the fact that the MCS needs to retransmit each AAL_SDU received from the senders. The interrupt frequency increases with the increase in number of senders. The MCS therefore takes a longer time to retransmit 20 MB. This in turn increases the delay for the receiver to get the 20 MB. Thus the experiment highlights the fact that a single MCS can easily be overloaded with over-active senders.

The second experiment was aimed at comparing the delays involved in using a multicast IP router (an *mrouter*) between 2 smaller LISs, as opposed to using an MCS with a larger LIS. In the first case, a sender to group X existed on LIS a.b.c.d, and a receiver (member of group X) existed on LIS a.b.e.f. The *mrouter*, which had interfaces on both LISs, enabled data from the sender to be delivered to the receiver. The experiment measured the average application

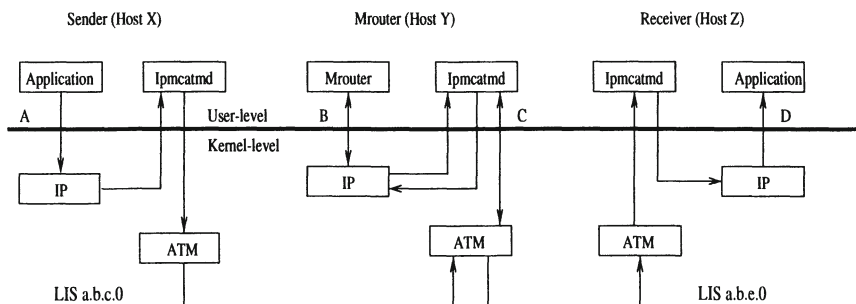


Figure 4 Use of an Mrouter with the MARS architecture.

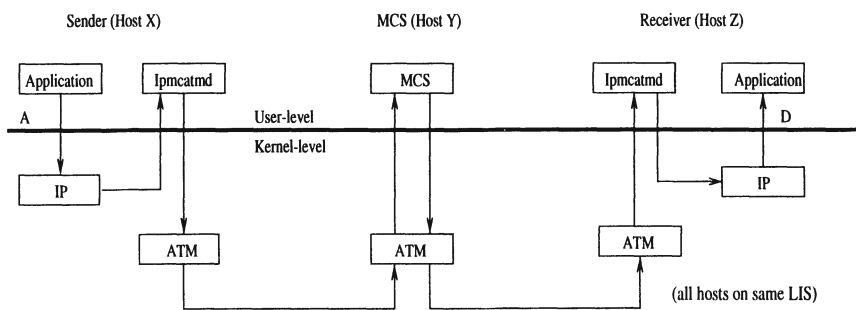


Figure 5 Use of an MCS.

level delay for receiving 2 bytes at the receiver which were transmitted by an application at the sender (delay from A to D in figure 4). The second case involved using the sender and receiver on one LIS, but having an MCS forward traffic from the sender to the receiver. The delay was measured as before (from A to D in figure 5).

An average delay of **16 msec** was observed in the mrouter case, while an average delay of **8.5 msec** was observed in the MCS case. The mrouter introduced significant additional delay in the data path since it required data to be passed through an additional IP layer (at the intermediate mrouter). Also, the processing on data packets done by an mrouter is more than that done by an MCS. High performance hardware that could be used to make an mrouter run faster would also have an equivalent effect on a high performance MCS. Whilst the numbers (16ms vs 8.5ms) are not important in absolute terms, they are interesting as relative measures. Thus use of MCS, which enables us to use larger LISs due to lower VC usage, is definitely preferable to use of VC Mesh, which due to higher VC usage may lead to smaller LISs (as explained in section ??), and hence needs more LISs interconnected by mrouter to accommodate the same number of hosts.

The above experiments conclude that although MCSs are better than mrouter from the performance perspective, they can become a bottleneck for groups

with a large number of senders. We now present mechanisms for supporting multiple MCSs, which can be a solution to the above problem.

5 MULTIPLE MCS ARCHITECTURES

The primary motivation for using more than one MCS to support an IP multicast group in the cluster is fault tolerance and load sharing. Fault tolerance is achieved as the remaining MCSs can take over the load of a failed MCS. Load sharing is achieved by using either sharing senders or group members.

The senders to a group can be shared amongst the MCSs. It is important to ensure that each sender is supported by one and only one MCS to avoid the possibility of duplicate data reception by group members. The term “supporting a sender” indicates that an MCS will forward data received from a sender. The point-to-multipoint data VC at each MCS will have all the group members as leaves. Sharing senders helps in alleviating the problem of the MCS being a bottleneck. Each MCS can now support the same or fewer number of senders than before, with the total number of senders supported being the sum of the senders supported by all the MCSs, and hence more than in the single MCS case.

The group members can be shared such that each group member receives data from one and only one MCS. Each MCS will receive and forward data from all the senders, but will maintain the outgoing VC to only those group members supported by it. Here the term “supporting a group member” indicates that an MCS will have only those group members supported by it as leaves of its outgoing VC. Sharing group members is better than sharing senders from the VC usage perspective. This is because each group member terminates a VC from each MCS in the latter case, while it terminates a VC from one MCS only in the former case. This is only as good as using a single MCS, hence sharing group members is useful only if fault tolerance using multiple MCSs is needed and the VC count is to be minimized.

These options, as well as the choice of using either the single or multiple MCS approach, or the VC Mesh approach, can be chosen on a per group basis in the cluster. We do not consider the option of permitting both senders and group members to be shared simultaneously because of the design complexity that such support will demand. We now present three mechanisms for the support of multiple MCSs.

5.1 Fault tolerance using multiple MCSs

A minimal approach for supporting multiple MCSs is offered by this mechanism. One or more MCSs can be designated to support an IP multicast group. One of the MCSs functions as the “active” MCS, in that it supports the group by forwarding data from senders to group members. The remaining MCSs function as backups, with a backup MCS starting support of the group

if the currently active MCS fails. Thus the mechanism offers fault tolerance but no load sharing, as only one MCS is permitted to be active at a time. The inter-MCS synchronization mechanism is simplified by determining the identities of all the MCSs designated to support a particular group prior to start up. This “list” also determines the order in which a new MCS is to be selected, in case of failure of the existing one. The MCSs use a heartbeat mechanism with HELLO messages.

An MCS on startup determines its position in the MCS “list”. An MCS registers for supporting the group with the MARS only if it is first in this “list”. The first MCS thus becomes the active MCS and supports the group as described in section 2.2. The active MCS also opens a point-to-multipoint VC to the remaining MCSs in the set (the inactive MCSs). It starts sending HELLO messages on this VC at a fixed interval (*HelloInterval* seconds). The inactive MCSs maintain a timer to keep track of the last received HELLO message. If an inactive MCS does not receive a message within *HelloInterval***DeadFactor* seconds (values of *HelloInterval* and *DeadFactor* are the same at all the MCSs), it assumes failure of the active MCS and attempts to elect a new one. Consensus amongst at least half of the existing inactive MCSs is needed before an MCS is elected as the new active MCS. This ensures that more than one MCS does not get elected, and also that deadlock is avoided. If an MCS is elected as the new active one, it registers to support the group with the MARS. It also initiates the transmission of HELLO messages to the remaining inactive MCSs.

The main advantage of the above mechanism is that it is conceptually simple and easy to implement. No exchange of information other than the heartbeat messages is needed for inter-MCS synchronization. The detailed description can be found in (1997), which is being considered by the ION group for release as an Informational RFC.

5.2 Fault tolerance and load sharing using multiple MCSs

Load sharing implies that multiple MCSs can actively support the group by sharing senders or group members. The need for load sharing capability however introduces additional complexity in the inter-MCS synchronization protocol. An allocation entity is needed which is responsible for allocating senders/group members amongst the MCSs. It needs to be a single (not distributed) entity so as to minimize complexity. This entity will be located at one of the MCSs (this is designated as the primary MCS, with the others being secondary), and the synchronization protocol will have to provide a recovery mechanism if the primary MCS fails.

The mechanism we propose uses an inter-MCS synchronization protocol to inform the MCSs about the senders/group members supported by each MCS. One MCS is designated as the primary MCS, and this MCS functions as the allocation entity. The remaining MCSs are designated as secondary MCSs.

An MCS registers to support a group with the MARS on startup. The MCS is told whether it is a primary or secondary MCS at startup. A primary MCS supports the entire group by itself when no other MCS exists. The primary MCS communicates with each secondary MCSs through a point-to-point VC. It first communicates with a secondary MCS when the secondary MCS starts up. If the secondary MCS does not agree with the primary MCS about sharing senders or group members on startup, the secondary MCS aborts after deregistering from the MARS. Otherwise, it is informed by the primary MCS about the senders/group members that are to be supported by it. If senders are being shared, the MCS forwards data from only those senders supported by it to all group members. Data received from other senders is dropped. If group members are being shared, the MCS opens a point-to-multipoint VC to only those group members supported by it, and forwards data from all senders on the VC. The primary MCS also provides the secondary MCSs with a list of other MCSs supporting the same group. This list is consistent across all the MCSs supporting the same group, and is updated by the primary MCS at a regular interval. Secondary MCSs use this list to select a new primary MCS in case of failure of the existing one. The possibility of electing more than one primary MCS, or of deadlock in the election process, is reduced as this list is consistent across all the MCSs. The newly elected primary MCS reallocates the senders/group members amongst the existing MCSs. The allocation mechanism also provides for reallocation of senders/group members that were being supported by a failed secondary MCS amongst the remaining MCSs. This reallocation is done by the current primary MCS. The primary MCS also allocates any new group members to an existing MCS. The protocol uses well-defined control messages (MCS_MULTI, MCS_LIST, MCS_REQUEST, MCS_HELLO, MCS_MULTIACK, MCS_SELECT) for inter-MCS synchronization, which are described in (1996d).

5.3 Fault tolerance and load sharing using an enhanced MARS

The previous mechanisms were designed for use with MARS servers and clients conforming to (1996a). No change was needed to either the MARS server or the clients. One of the MCSs was actually used as the allocation entity in the second proposal. However, the MARS is a repository of all relevant information about the cluster (e.g., identity of senders/members of all groups, MCSs in the cluster, groups supported by each MCS, etc). It has access to all the information that an allocation entity might require, and thus blends naturally into the role. Using the MARS in this way also removes the need for any inter-MCS synchronization. This is because the allocation and consequent load sharing can take place transparent to the MCSs supporting a group, with the MARS making all the decisions and conveying information to cluster

members after appropriate processing. We propose such an inclusive approach for supporting multiple MCSs using the MARS server.

Individual MCSs are ignorant of the other MCSs supporting the same group. This simplifies their design, making it sufficient for them to conform to the basic MCS architecture described in (1997). Another advantage is that MCSs do not need to receive (and drop) data from senders not supported by them (as happens in the mechanism described in section 5.2). This is because the MARS server can selectively inform each sender about the identity of the MCS supporting it. The enhanced MARS approach is described in detail in (1996e).

6 SUMMARY

The MCS approach is demonstrated to be better than the VC Mesh approach in terms of VC usage, number of groups permissible in the LIS, and signaling load. Using an MCS is also shown to be better from the performance perspective than using IP multicast routers with the VC Mesh approach. However, a single MCS is demonstrated to be a bottleneck, especially for groups with a large numbers of senders. This problem, and the related problem of a single MCS being a central point of failure, can be resolved through the use of multiple MCSs. Three mechanisms are proposed which offer varying levels of multiple MCS support. These mechanisms have been specified as Internet Drafts (1997, 1996d and 1996e), respectively), and have been presented to the ION group for consideration as standard documents describing MCS architectures. The enhancements to the MARS discussed in section 5.3 have been suggested to the ION group for inclusion in the next version of the MARS specification.

REFERENCES

- [1] Almeroth, K. and Ammar, M.H. (1996), Collection and Modeling of the Join/Leave Behavior of Multicast Group Members in the Mbone, *High Performance Distributed Computing Focus Workshop (HPDC '96)*, Syracuse, New York, August 1996.
- [2] Armitage, G.J. (1996a), Support for Multicast over UNI 3.0/3.1 based ATM networks, *RFC 2022*, November 1996.
- [3] ATM Forum (1995), ATM User Network Interface (UNI) Specification Version 3.1, ISBN 0-13-393828-X, Prentice Hall, Englewood Cliffs, NJ, June 1995.
- [4] Online public release of MARS in part source, part object code form (1996c). URL: <ftp://ftp.bellcore.com/pub/gja/mars>, 1996.
- [5] Talpade, R., and Ammar, M.H. (1996d), Multicast Server Architectures for UNI 3.0/3.1 based ATM networks, *Internet Draft, draft-talpade-ipatm-marsmcs-01.txt*, March 1996.
- [6] Talpade, R., and Ammar, M.H. (1996e), Multiple MCS support using an enhanced version of the MARS server, *Internet Draft, draft-talpade-ion-multmcs-01.txt*, December 1996.
- [7] Talpade, R., and Ammar, M.H. (1997a), Multicast Server Architectures for UNI 3.0/3.1 based ATM networks, *Internet Draft, draft-ietf-ion-marsmcs-02.txt*, January 1997.

Center Placement Algorithms for Large Multicast Groups

A. S. Weigmann and J. Nonnenmacher and E. W. Biersack
Corporate Communications Department, Institut EURECOM
B.P. 193, 06904, Sophia Antipolis Cedex, FRANCE
{nonnen, erbi}@eurecom.fr

Abstract

An increasing number of distributed applications require a specific form of multicast called dissemination, in which a single source *reliably* transfers data to multiple receivers. Reliability leads for large groups of receivers (100s or 1000s of participants) to the problem of feedback implosion at the source and to a decrease of transmission efficiency. The cluster approach was identified to have excellent scalability with the number of receivers. It partitions the multicast delivery tree into clusters, where a representative in the cluster called center is used for local feedback processing and local transmission. Up to now, clustering/center placement has been done administratively or based on network addresses. Needed are center placement algorithms, allowing the introduction of placement criteria based on the network topology and on delay. In this work, three center placement algorithms designed for static multicast groups are presented and simulation results are shown in order to assess their performance.

Keywords

Multicast, Feedback Implosion, Local Retransmission, Center Location

1 INTRODUCTION

Emerging high speed networks and the widespread availability of multimedia capable workstations have drastically increased the demand for distributed applications. Along going with this evolution comes an increasing need for multicast, allowing to distribute data to many users while efficiently using scarce communication bandwidth.

Many applications require a specific form of multicast called dissemination [1, 19, 2, 20], in which a single source *reliably* transfers data to multiple receivers. Applications include electronic newspaper distribution, WWW in-advance caching, file transfers and shared whiteboard (see also [1, 13]).

While some applications, like the shared whiteboard, support *dynamic* multicast groups,

where receivers dynamically join or leave the group during transmission time, numerous other ones, like newspaper distribution or file transfers, are designed for *static* groups of receivers, which remain unchanged throughout the entire transmission.

IP Multicast [4], which forms the basis of the MBone [6, 16] provides scalable, but not reliable multicast. Reliability requires that either data packets are acknowledged by the receivers (ACKs) or that packet losses are reported using negative acknowledgments (NACKs). For larger groups of receivers (100s or 1000s of participants), this leads to the problem of packet implosion at the source [15]: The source suffers from the significant overhead incurred due to the processing of the ACKs or NACKs from the receivers.

The research community proposed two directions to deal with the scalability problem:

- *timer approaches* [8, 7] that introduce a certain asynchrony among receivers to protect the source from an implosion and
- *cluster approaches* [20, 13, 9] that achieve the same goal by introducing a hierarchy into the multicast tree.

In the cluster based approaches, the multicast delivery tree is portioned into **clusters**. Every cluster has a representative, the **center**. Receivers do not acknowledge received packets directly to the source but only to the center of their cluster.

All centers are organized in a hierarchy. A center is responsible for all receivers in its cluster as well as for other centers located one level below in the cluster hierarchy. A center receives ACK packets from receivers and centers it is responsible for and forwards a single acknowledgment packet to the center in the next higher hierarchy level (ACK accumulation).

Most of the cluster based approaches presented so far either use centers placed administratively in the network [13] or rely on centers placed algorithmically *within* a group (cluster), which is either defined by hand or based on network- (subnet-) addressing [20, 11]. Administratively placed centers or clusters do not scale well for large groups of receivers and clustering based on addressing does not reflect the real network structure (connectivity and delay relations) but only the address structure.

Needed are center placement algorithms, allowing the introduction of placement criteria based on the network topology and on delay relations and not only on addressing schemes. In this work, criteria for a “good” center placement are stated. Three center placement algorithms designed for static multicast groups are presented and simulation results are shown in order to assess their performance.

The remainder of this paper is organized as follows. In section 2, the center placement problem is described in a formal way. In section 3, placement algorithms are presented and section 4 shows simulation results. In section 5, final conclusions are drawn.

2 THE CENTER PLACEMENT PROBLEM

The problem of placing centers in a multicast tree can be looked at from a graph-theoretical point of view. This allows to approach the problem without the need to consider implemen-

tational aspects, which are linked very closely to a specific type of network. Once the center placement problem is solved in theory it can be transferred to reality and implemented.

The graph-theoretical point of view abstracts from the real connection endpoints (receivers) and considers just multicast routers. Regarded is a directed **tree** (routers = vertices, links = edges) rooted at the multicast source. Edge weights represent link delays. **Receivers** can be located on every vertex except on the tree's root. All leaves of the tree are receivers, but vertices that are not leaves can also be receivers.

Centers can be placed on all internal vertices, thus not on leaves. The root is implicitly a center. The **number of centers** in the tree is called **NOC**.

The criteria for center location are based on a given **optimal cluster size Z** and the minimization of the end-to-end delay between the source S and the receivers $\in R$ in a multicast tree T .

The following definitions help to specify the criteria for center placement.

Definition 1 Directly Served

A receiver r_k (a center c_j) is **directly served** by a center c_i when no other center is located on the path in the tree T between c_i and r_k (between c_i and c_j). c_i is not directly served by itself.

Definition 2 Cluster with Center c_i

The **cluster with center c_i** is defined as the set of all the receivers and centers directly served by c_i , c_i itself, and all the edges interconnecting the elements of this set.

Definition 3 Cluster Size $cs(c_i)$

The **cluster size $cs(c_i)$** of the cluster with center c_i is the number of centers and receivers directly served by c_i .

2.1 Cluster size

The cluster size plays a major role in the success of reliable multicast with clustering. The amount of feedback a center c_i is capable to process imposes a scalability limit on the cluster size. Also the number of packets a center has to retransmit for its cluster increases with the number of receivers or further centers directly served by c_i and has a major influence on the performance.

In order to limit the cluster size of any cluster with center $c_i \in C$ to a given optimal cluster size Z , a **cluster size constraint (CSC)** will be introduced:

$$\forall c_i \in C : cs(c_i) \leq Z \quad (1)$$

The **mean squared error of cluster sizes MSE** is defined as

$$MSE = \frac{1}{NOC} \sum_{i=1}^{NOC} (Z - cs(c_i))^2 \quad (2)$$

The minimization of the *MSE* will lead to a optimal center placement with respect to a given optimal cluster size Z . Minimizing the *MSE* is the first criterion for center placement and is referred to as the *Cluster Size Criterion*.

2.2 End-to-End Delay

In multicast connections the end-to-end delay has a big influence on the performance of a reliable transport protocol. At every center c_i feedback from directly served centers and receivers must be processed. The end-to-end delay between a receiver r_j and the source S is composed of the delay of the links the packet transverses and a number of **center delays** cd experienced at every center. The cumulative delay is the **receiver delay** $D(r_j)$, defined as the sum of the edge delays $d(v_k, v_l)$ on the path from the source S to a receiver r_j in the tree T plus the product of the number of centers on the path and a constant delay cd experienced at every center. The end-to-end delay of the whole multicast connection is defined as the **maximal receiver delay** mDR of all r receivers:

$$mDR = \max_{j=1..r} \{D(r_j)\} \quad (3)$$

The objective, referred to as the *Delay Criterion*, is to minimize the delay mDR . Please note, that the objective is not the same as to minimize the maximal number of centers on the path from the source to a receiver. Having an arbitrary center placement, the path to the receiver that yields mDR may be different from the path where the most centers can be found on.

3 CENTER PLACEMENT ALGORITHMS

Static multicast trees occur to be important in the context of reliable multicast as there exist numerous applications like newspaper distribution, WWW in-advance-caching and software distribution. All three examples have in common that the group of receivers is determined preceding the multicast transmission and that no further receivers will join or leave this group after the transmission has started.

Static multicast trees are also the easier case, as the center placement can be done in advance. The placement is done before the actual multicast transmission starts and is not time critical. Also extensive exchange of status information (for example information on the tree structure) between nodes would be possible (although not desirable).

In the following sections, three placement algorithms are presented:

- Center Placement by Split and Shift (CPSS)
- Center Placement by Shift and Merge (CPSM)
- Center Placement by Dynamic Programming (CPDP)

Two cases will be considered for every algorithm:

1. **Placement with cluster size constraint (CSC):** Center placement is done with respecting the cluster size constraint (CSC) defined in section 2.1.
Note that if a center is placed on a vertex with an outdegree* greater than Z , the cluster size constraint cannot be met as the smallest feasible cluster size equals the outdegree.* Anyway, this case is only a theoretical one. In practice, and also in the trees used for simulations, the maximal outdegree remains below the optimal cluster size Z .*
2. **Placement without cluster size constraint (CSC):** The cluster size constraint CSC is dropped. This allows more flexibility for center placement at the expense of a non-deterministic maximal cluster size. Note that only the cluster size constraint is removed – the objective to minimize the mean squared error of the cluster size remains valid.

Before moving on to the different placement algorithms, two more expressions are defined. Although the algorithms can also be implemented without them, they allow to increase the efficiency in terms of execution speed of the algorithms both in the simulation environment and in a real implementation.

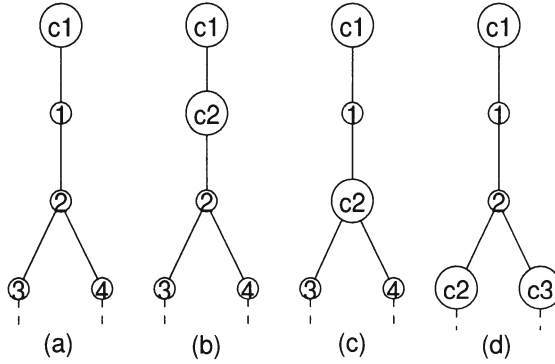


Figure 1: Furthest Directly Connected Descendant (FDCD). (a) Original tree, all nodes are routers (no receivers), the root is implicitly a center, (b) center on node 1: $cs = 1$, (c) center on node 2: $cs = 1$, (d) center on nodes 3 and 4: $cs = 2$.

Assume the following situation (see also Figure 1): A center c_i is located on a vertex v_i . Centers directly served by c_i are located on descendants of v_i . It can be seen that the size $cs(c_i)$ of the cluster with center c_i changes, if a directly served center is shifted from a vertex v to its children, if (i) vertex v has an outdegree > 1 ; or (ii) it is a receiver. In order to formalize this effect, **directly connected nodes** and **furthest directly connected descendants (FDCDs)** are defined:

*The number of children of a node x in a tree T is called the *outdegree* of x .

*All children of the vertex the center is placed on are either leaves or internal vertices with centers.

*Work done by the Routing community considers average outdegrees between 2 and 8 [18] and maximal outdegrees of 4 and 5 [12] for networks. The outdegree for multicast trees is typically smaller than the outdegree in the network. The optimal value Z is not known yet; it will be in the order of 20–30.

Definition 4 Directly Connected

A vertex $v_j \in V_t$ located in a tree $T = (V_t, E_t)$ is defined as being **directly connected** to an ancestor v_i if there are no other vertices with an outdegree > 1 , centers or receivers on the path between v_i and v_j .

Definition 5 Furthest Directly Connected Descendant (FDCD)

In a tree $T = (V_t, E_t)$ a directly connected descendant $v_j \in V_t$ of a vertex $v_i \in V_t$ is called a **furthest directly connected descendant (FDCD)** of v_i if there exists no other directly connected vertex $v_k, k \neq j$ with a path between v_i and v_k that includes the path from v_i to v_j .

3.1 Split and Shift (CPSS)

The first algorithm called Center Placement by Split and Shift (CPSS) belongs to the group of *greedy* algorithms [3]. *Greedy* algorithms always make the choice that looks best at the moment. That is, they make a locally optimal choice in the hope that this choice will lead to a globally optimal solution.

CPSS starts the center placement by building the first cluster with its center located on the root of the tree. It places directly served centers on the root's furthest directly connected descendants (FDCDs) and subsequently tries to *shift* them towards the leaves. Centers are always shifted to their furthest directly connected descendants (FDCDs). Centers that must be shifted and that are located on a node with an outdegree > 1 are *split* and all FDCDs of this node become centers.

Once the cluster is created, the algorithm is repeated recursively for all placed (directly served) centers.

The directly served centers are shifted due to the following rules:

1. Minimize the maximal receiver delay mDR .
2. If the delay cannot be optimized any further, try to optimize the size of the cluster.

Minimizing the maximal receiver delay mDR does not imply that the total number of centers in the tree has to be as small as possible. It means that the number of centers on the longest path (in terms of delay) has to be as small as possible, as every center on the path to a receiver causes an additional center delay cd . The longest path *before* and the longest path *after* the center placement do not necessarily have to correspond.

In order to keep the number of centers on a certain path as small as possible, the centers on this path must be located as far apart from each other as possible. Following this rule, CPSS always tries to shift the center on the longest path towards the leaves. If a center is shifted onto a leaf, it will be removed since centers located on leaves do not make sense (they cannot serve any other node).

CPSS does not shift the center on the longest path if the cluster size constraint (CSC) is violated. In this case, CPSS tries to shift the center on the second longest path. If this one cannot be shifted either, it tries to shift the one on the third longest path and so on. If it

does not succeed in shifting any center, the cluster is “finished” and the clusters for the just placed directly served centers can be built.

3.2 Shift and Merge (CPSM)

The second algorithm presented, called Center Placement by Shift and Merge (CPSM), also belongs to the group of *greedy* algorithms [3]. Unlike the CPSS algorithm, CPSM starts building clusters at the leaves and then moves subsequently towards the source. It concentrates on optimizing the cluster sizes (and thereby the mean squared error of the cluster size MSE). Less effort is spent in minimizing the maximal receiver delay mDR since CPSM does not explicitly try to reduce the number of centers on the longest path (in terms of delay), like CPSS does.

The first cluster is built by randomly choosing a leaf in the multicast tree and placing a center on this node (Figure 2 (a) and (b), node 4).

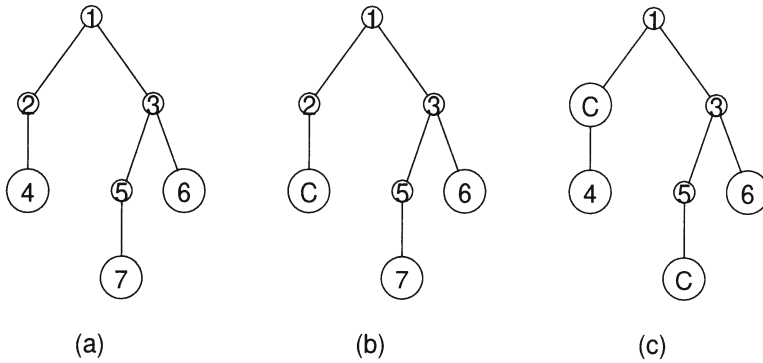


Figure 2: Center Placement by Shift and Merge (CPSM), part 1: (a) Tree without centers. Big circles represent receivers, small circles represent routers, (b) placing center on leaf, (c) shifting center towards root and placing new center on leaf in sibling’s subtree.

In the following steps, this center will be *shifted* towards the root until it is located on a vertex that has one or more siblings* (node 2 in the example). Now, in every subtree rooted at one of these siblings, a center has to be placed on a randomly chosen leaf (Figure 2 (c), node 7), which is then again propagated towards the root. If a center encounters a node with one or more siblings (node 5), again, there has to be placed a center on a randomly chosen leaf (node 6) in every subtree rooted at a sibling and so on (Figure 3 (a)).

This leads to a situation, where a vertex (node 3) has two or more children a center is located on. Now a selection process starts, deciding, which of the centers are shifted to the parent node. First, this decision is made for the center on the longest path (in terms of

*Two nodes having the same parent are called **siblings**.

delay), then for the center on the second longest path, then for the one on the third longest path etc. Note, that already placed centers contribute to the length of a path. If more than one center is shifted, the centers will be *merged* to a single center (Figure 3 (b)). If no center

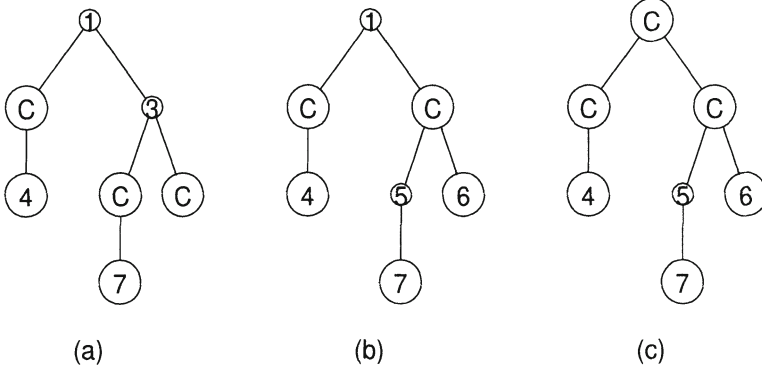


Figure 3: Center Placement by Shift and Merge (CPSM), part 2: (a) Shifting center towards root and placing new center on leaf in sibling's subtree, (b) two centers have been shifted to their parent and have been merged, (c) no center has been shifted, a new center will be placed on the parent.

is shifted, a new center will be placed on the parent (Figure 3 (c), node 1).

The rules for deciding, which centers are moved, are the following:

- A center located on a leaf is always moved.
- If the new cluster size $cs_{new}(\text{parent}(V))$ after shifting a center on node V to its parent is closer to the optimal value Z than the cluster size $cs(V)$ of V before the shifting and not higher than Z , the center will be moved.
- If the cluster size $cs_{new}(\text{parent}(V))$ after shifting a center on node V to its parent is equally close to the optimal value Z as the current cluster size $cs(V)$ (without shifting) and if V lies on the path from parent(V) to the node in the subtree of parent(V) yielding the mDR or no center has been shifted to the parent of V yet, the center will be shifted.
- In all other cases, the center will not be shifted.

3.3 Dynamic Programming (CPDP)

The third algorithm presented is based on the method of dynamic programming, which is related to *divide-and-conquer* [3] and *branch-and-bound* [14] in the sense that it partitions a problem into subproblems and *intelligently* enumerates the feasible points of a combinatorial optimization problem.

To solve the problem, thus to find the best placement for a tree, dynamic programming

is used in combination with a *local search*. Supposed, the best placements for *all* subtrees, as well as all important parameters of these placements, like the number of clusters in each subtree, the *mDR* in each subtree, etc. are known. Then, the cluster with its center located on the tree's root is built simply by placing directly served centers systematically on the root's descendants and by evaluating the different combinations. In practice, this means: Centers are placed on the root's furthest directly connected descendants (FDCDs) and then propagated towards the leaves. At every node with outdegree > 1 , centers are split. Evaluation of the different center placements is done using a metric composed of the relative increase of the maximal receiver delay *mDR* and the normalized mean squared error of the cluster sizes (*MSE*):

$$Metric = 0.5 \cdot \frac{mDR_{with\ centers} - mDR_{without\ centers}}{mDR_{without\ centers}} + 0.5 \cdot \frac{MSE}{Z^2} \quad (4)$$

Once, the optimal placement for a (sub-)tree has been found, the center configuration as well as the parameters characterizing the placement, like the squared errors of the cluster size, the number of clusters, the *mDR*, etc. will be stored allowing them to be reused as the result of a solved problem.

In order to find the best placement, all feasible combinations must be evaluated and the number of feasible combinations is growing exponentially with the number of nodes in the tree. In order to limit the search to a bearable number of combinations, a **maximal search depth without improvement *MSD*** is introduced. If a certain center has been propagated for *MSD* steps without experiencing any improvement in the quality of the placement in terms of cluster size and delay, this center will be abandoned and not propagated any further. The introduction of this limit restricts the number of evaluated combinations, thereby limiting computational complexity, but also excluding numerous combinations. The optimal solution might not be found, the result has sub-optimal character.

4 RESULTS

In the last sections, three center placement algorithms have been presented. In order to evaluate their performance, they were implemented and executed on a number of networks.

As input data, 10 networks with 200 nodes, an average outdegree of 3.0 and an average link delay of 412.38 were created following Doar [5], who modified the method of Waxman [17] to avoid an increasing outdegree for an increasing number of nodes. The network construction of Waxman/Doar is commonly used by the Multicast Routing community for comparing the performance of different multicast routing algorithms.

For each of the 10 networks, 10 different trees with 10, 20, 40, 60, 80, 100, 120 and 140 receivers each on randomly chosen locations have been constructed using a shortest path tree (SPT) algorithm [5]. This means that for a given number of receivers, 100 trees have been created, which results in a total of 800 trees.

The three center placement algorithms CPSS, CPSP and CPDP were executed on every SPT:

- *with* respecting the cluster size constraint (with CSC)
- *without* respecting the cluster size constraint (without CSC)

For the optimal cluster size Z the values 5, 10, 15, 20, 25 and 30 have been used, for the center delay cd the values 100.0, 200.0 and 300.0, resulting in 18 different parameter sets.

All values presented in the following sections are mean values of 100 evaluated trees. Indicated intervals are 95% confidence intervals [10].

4.1 Evaluation Metrics

In order to compare different algorithms, an exact quantification is needed. As the effects of the cluster size and the maximal receiver delay to a multicast connection are not known exactly, they will be treated separately. For the performance evaluation two metrics are used. The quality of the placement with respect to the cluster size is measured by the normalized MSE , defined as cluster size error CSE :

$$CSE = \frac{\sum_{i=1}^{NOC} (Z - cs(c_i))^2}{Z^2 \cdot NOC} \quad (5)$$

The quality of the center placement with respect to delay is measured by the delay increase cost DIC representing the relative increase of the maximal receiver delay due to the center placement:

$$DIC = \frac{mDR_{with\ centers} - mDR_{without\ centers}}{mDR_{without\ centers}} \quad (6)$$

In the next sections, the proposed algorithms will be evaluated using the above introduced quality metrics.

4.2 Split and Shift (CPSS)

(a) Number of Centers NOC

Figure 4 shows the distribution of the distances in hops between the tree's root and the centers for 40 receivers trees, $Z = 20$ and $cd = 100.0$. It clearly shows that CPSS tends to locate centers further away from the root than do CPSM and CPDP. Note that the distance 0 is caused by the center located on the root. The lower normalized count for distance 0 for CPSS in contrast to CPSM and CPDP results from the higher number of centers for CPSS in general (see also Figure 8).

Figure 5 shows for CPSS the number of centers that lie on the paths from the root to the receivers for 40 and 140 receivers trees ($Z = 10$, $cd = 100.0$).

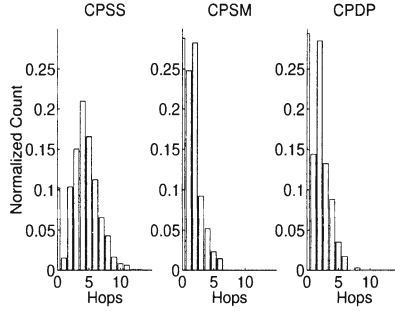
Dist. Source-Centers (Hops), with CSC, 40 Recv, $Z=20$, $CD=100$ 

Figure 4: Distribution of distances from source to centers in hops. CPSS, CPSM and CPDP with cluster size constraint

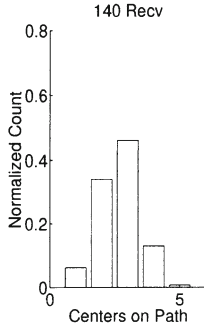
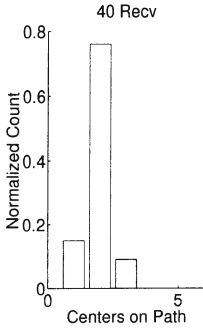
Centers on Paths to Recv, CPSS (with CSC), $Z=10$, $CD=100$ 

Figure 5: Centers on the paths to receivers, CPSS, with cluster size constraint, 40 and 140 receivers.

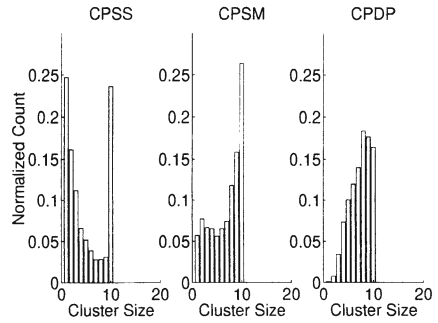
Cluster Size Distribution, with CSC, 140 Recv, $Z=10$, $CD=100$ 

Figure 6: Cluster size distribution for CPSS, CPSM and CPDP with cluster size constraint.

(b) Cluster Size Error CSE

CPSS suffers from the fact that numerous clusters near the leaves are sparsely filled. Figure 6 shows the cluster size distributions for the three algorithms CPSS, CPSM and CPDP with cluster size constraint (CSC) for 140 receivers and an optimal cluster size of $Z = 10$ ($cd = 100.0$). Conspicuous are the shapes of the distributions. CPSS often succeeds in reaching the optimal cluster size Z . The high portion of sparsely filled clusters emerges from those clusters located in the leaves areas of the trees. For 140 receivers, $Z = 10$ and

$cd = 100.0$, 63.7% of all clusters created by CPSS have a size of $cs(c_i) \leq 5 = \frac{Z}{2}$ (for comparison: CPSM: 32.2%, CPDP: 21.6%).

(c) Delay Increase Cost *DIC*

In section 2, it has been pointed out that the maximal receiver delay mDR plays a key role in the performance of reliable multicast connections. The relative increase of the mDR (6) is regarded as the basic criterion for evaluating the quality of the placement algorithms in terms of delay.

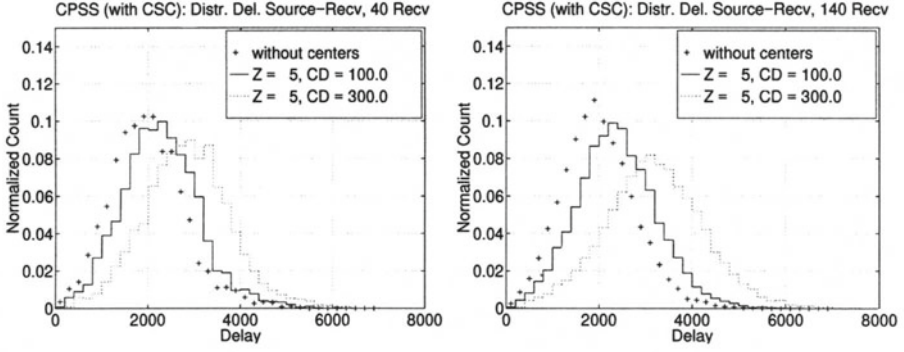


Figure 7: Receiver delay distribution before and after center placement for CPSS with cluster size constraint for 40 and 140 receivers

Figure 7 shows the impact of centers on the general distribution of receiver delays: * The delays are increased (the shape is moved to the right) and the delays are distributed more equally (flatter shape).

4.3 Shift and Merge (CPSM)

Center Placement by Shift and Merge (CPSM) starts placing centers at the leaves of the tree and concentrates primarily on optimizing the cluster size. Consequently, the performance in terms of the cluster size error CSE is very good and outperforms the Center Placement by Split and Shift (CPSS), while the performance in terms of delay is lower than the one of CPSS.

(a) Number of Centers *NOC*

In contrast to CPSS, CPSM avoids the effect of sparsely sized clusters (see Figure 6). Building clusters from the leaves to the root results in only one really badly sized cluster, the one

*In the figure, receiver delays have been put into bins with a width of 200.0

located at the tree's root. As the average cluster size is closer to the optimal value Z , the total number of clusters in a tree is less than for CPSS (Figure 8). Note, that the number of centers (NOC) in the trees grows approximately linear with the number of receivers $NOC \approx k \cdot r$ with a gradient k inversely-proportional to the optimal cluster size Z : $k \sim \frac{1}{Z}$, $\frac{r}{Z} \gg 1$.

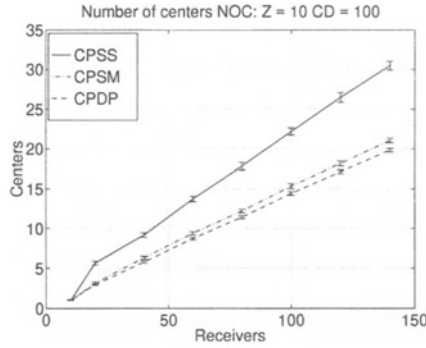


Figure 8: Number of centers NOC . CPSS, CPSM and CPDP with cluster size constraint.

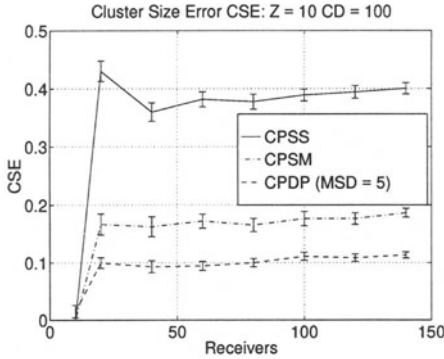


Figure 9: Cluster Size Error CSE . CPSS, CPSM and CPDP with cluster size constraint.

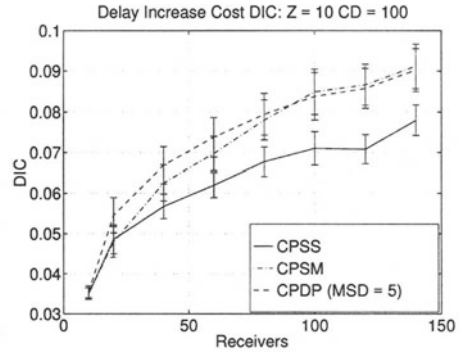


Figure 10: Delay Increase Cost DIC . CPSS, CPSM, CPDP with cluster size constraint (CSC).

As can be seen in figure 9 performs CPSM much better than CPSS in terms of the cluster size error CSE . CPSM prefers to merge clusters instead of just expanding clusters along

the longest path (in terms of delay), like CPSS does.* This is in harmony with the objective of the algorithms: While CPSM optimizes the mean squared error of the cluster size (*CSE* criterion), CPSS tries to optimize the delay criterion (*DIC*), this can also be noticed in the better delay performance of CPSS (Figure 10). In contrast to the *CSE* criterion, which offers approximately constant values for different tree sizes, the *DIC* criterion produces lower values for smaller trees than for bigger ones. This effect cannot be attributed to the algorithms, but is the result of the testing environment: Random networks with $|V| = 200$ nodes have been used for the simulations. The larger the trees get, the higher the ratio receivers/nodes becomes. In total, more centers are placed in the trees, also on the *mDR*-paths, and the performance is decreasing.

4.4 Dynamic Programming (CPDP)

CPDP uses the method of dynamic programming. Instead of performing a *full* search on all nodes of the multicast tree for finding the optimal center locations, the amount of time needed to find a good solution is reduced by performing a *local* search, not considering combinations that are unlikely to lead to a good result.

The next paragraphs will show that there exists a close correlation between the extent of the local search and the quality of the resulting placement.

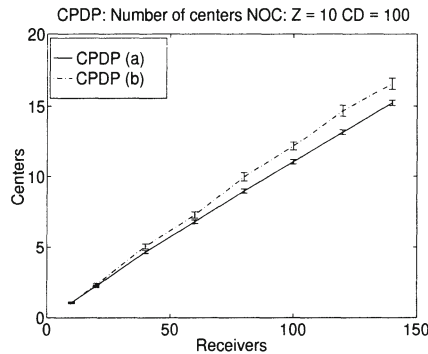


Figure 11: Number of centers *NOC*. Center Placement by Dynamic Programming (CPDP), different weights for the two criteria

(a) Number of Centers *NOC*

Comparing CPSM and CPDP (Figure 8), a slightly lower number of centers can be observed for CPDP. This little difference needs no further explanation, since the exact number of

*Note that this strategy can also be altered easily (for example, in order to put more emphasis on the delay criterion) by modifying the move-rules described in section 3.2.

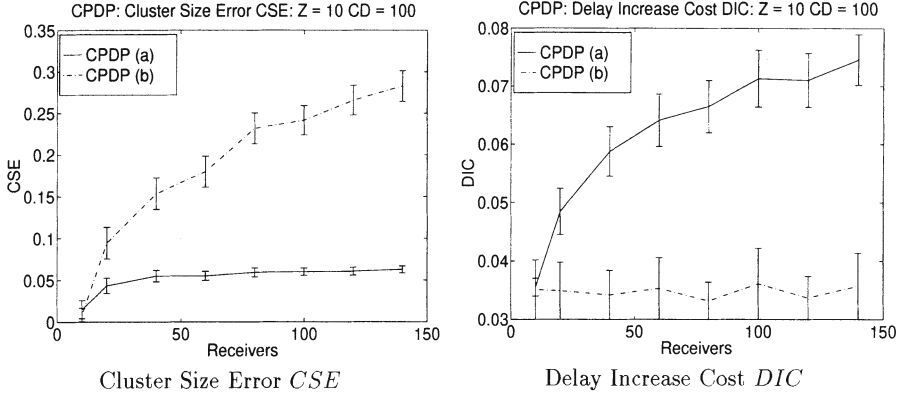


Figure 12: Cluster Size Error CSE and Delay Increase Cost DIC . CPDP (a) and CPDP (b), without cluster size constraint

centers placed by CPDP depends on the placement metric and thus might differ in both directions when the weights of the different criteria are altered.

CPDP's optimization function can be tailored by weighting the two criteria, cluster size and the delay. While CPDP (a) in Figure 11 weights both criteria equally (0.5 and 0.5), CPDP (b) uses a weight of 0.9 for the delay criterion and only 0.1 for the cluster size criterion. Giving more emphasis to the delay criterion results in a slightly higher number of clusters, a higher cluster size error CSE and a much lower delay increase cost DIC (Figure 12).

(b) Cluster Size Error CSE

Figure 13 shows the quality of the placement for CPDP without cluster size constraint (CSC) in terms of the cluster size error CSE for $Z = 5, 10, 20, 30$. We see a high correlation between the optimal cluster size Z and the required maximal search depth without improvement, MSD . While for $Z = 5$ $MSD = 2$ suffices* (Figure 13 (a)), there exists already a big penalty in the quality between a MSD of 2 and 4 for $Z = 10$ (Figure 13 (b)). Conspicuous is the fact, that for $Z = 10$ (Figure 13 (b)) the mean squared error of the cluster size in the trees with 10 receivers does not equal 0, which would be indicated by a CSE value of 0. This shows that the best solution, the one with all 10 receivers in one cluster of size 10, is not always found if the maximal search depth MSD is chosen too low.

*There is about no difference between the two lines for $MSD = 2$ and $MSD = 4$.

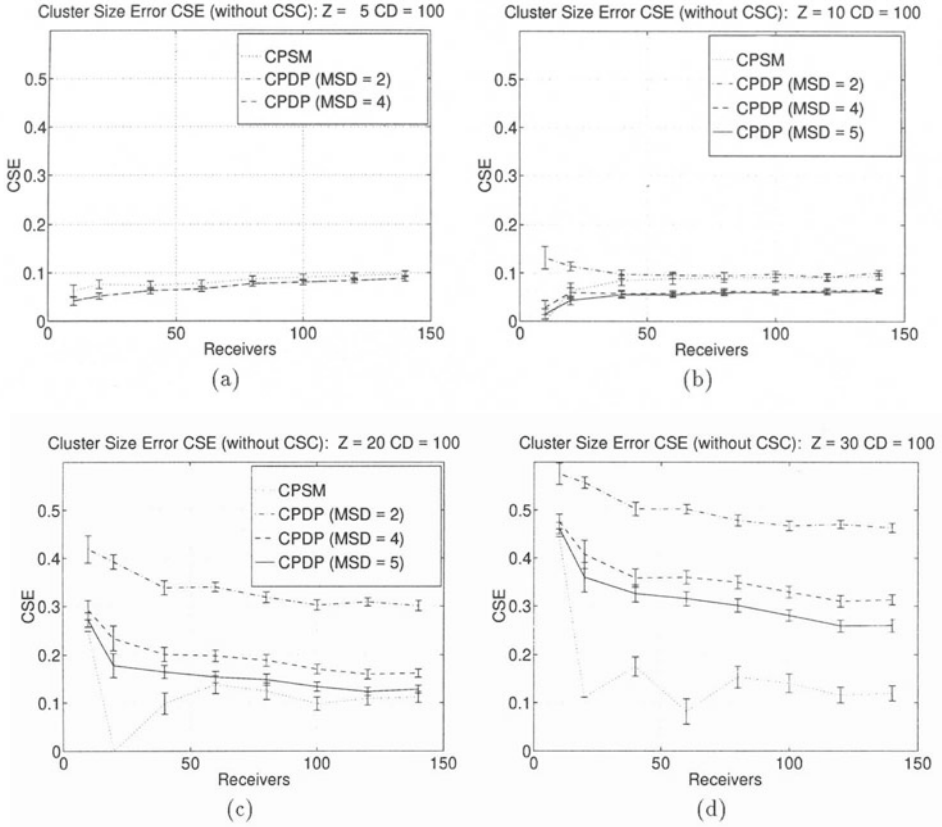


Figure 13: Cluster Size Error CSE for CPSM and CPDP without cluster size constraint

5 CONCLUSIONS

In the last sections, three center placement algorithms have been presented and evaluated. Two of them (CPSS and CPSM) start placing centers on either end of the tree and then proceed consecutively to the other end. The third one (CPDP) uses dynamic programming combined with a local search. CPSS performs very well in terms of delay, but does not try to optimize the cluster sizes resulting in a high cluster size error CSE . CPDM performs very well in terms of the cluster size error CSE , its delay increase cost DIC is worse than the one of CPSS. As long as the maximal search depth MSD is chosen high enough, CPDP has a good performance, both in terms of delay and cluster size, as its optimization function can be altered very easily to meet the requirements. However, CPDP has a high computational

cost which grows exponentially with the optimal cluster size Z . The results obtained lead to the following conclusions:

- For small optimal cluster sizes ($Z \in [5 \dots 10]$), CPDP is recommended, since its optimization function can be altered easily to meet predefined requirements.
- For bigger values of Z , CPDP's complexity is too high and preference should be given to CPSS or CPSM.
- If delay aspects have highest priority and cluster sizing only plays a minor role, CPSS should be used.
- If cluster size is important, CPSM should be used as it obtains small values for CSE and has a low computational cost.

REFERENCES

- [1] Azer Bestavros. Speculative data dissemination and service to reduce server load, network traffic and service time for distributed information systems. In *Proceedings of ICDE'96: The 1996 International Conference on Data Engineering*, New Orleans, Louisiana, USA, March 1996.
- [2] D. R. Cheriton. Dissemination-oriented communication systems. DSG Working Paper, 1992.
- [3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.
- [4] S. Deering. Host extensions for ip multicasting. *Internet Request for Comments*, RFC 1112, August 1989.
- [5] M. Doar and I. Leslie. How bad is naïve multicast routing. In *Proceedings of INFOCOM'93*, volume 1, pages 82–89. IEEE, 1993.
- [6] Hans Eriksson. MBONE: The multicast backbone. *Communications of the ACM*, 37(8):54–60, August 1994.
- [7] Sally Floyd, Van Jacobsen, Steven McCanne, Ching-Gung Liu, and Lixia Zhang. A reliable multicast framework for light-weight sessions and application level framing. In *Proceedings of SIGCOMM'95*, volume 25, pages 342–356. ACM, October 1995.
- [8] Matthias Grossglauser. Optimal deterministic timeouts for reliable scalable multicast. In *Proceedings of INFOCOM'96*, volume 3, pages 1425–1432, San Francisco, CA, USA, March 1996. IEEE.
- [9] Markus Hofmann. A generic concept for large scale multicast. In *International Zürich Seminar*, Zürich, Switzerland, February 1996.
- [10] Robert V. Hogg and Allen T. Craig. *Introduction to Mathematical Statistics*. Macmillan Publishing Co., Inc., New York, 1989.
- [11] H. W. Holbrook, S. K. Singhal, and D. R. Cheriton. Log-based receiver-reliable multicast for distributed interactive simulation. In *Proceedings of SIGCOMM'95*, volume 25. ACM, October 1995.
- [12] Vachaspathi P. Kompella, Joseph C. Pasquale, and George C. Polyzos. Multicasting for multimedia applications. In *Proceedings of INFOCOM'92*, pages 2078–2085. IEEE, 1992.

- [13] John C. Lin and Sanjoy Paul. Rmtp: A reliable multicast transport protocol. In *Proceedings of INFOCOM'96*, volume 3, pages 1414–1424, San Francisco, CA, USA, March 1996. IEEE.
- [14] Ch. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.
- [15] Sridar Pingali, Don Towsley, and James F. Kurose. A comparison of sender-initiated and receiver-initiated reliable multicast protocols. In *Proceedings of Sigmetrics, Conference on measurement and modeling of Computer Systems*, pages 221–230, Santa Clara, CA, USA, May 1994. ACM.
- [16] Macedonia. M. R. and D. P. Brutzmann. Mbone provides audio and video across the internet. *IEEE Computer*, April 1994.
- [17] B. M. Waxman. Routing of multipoint connections. *IEEE JSAC*, 6(9):1617–1622, December 1988.
- [18] Liming Wei and Deborah Estrin. The trade-offs of multicast trees and algorithms. In *Proceedings of ICCCN'94*, San Francisco, CA, USA, Sept 1994.
- [19] T. Yan and H. Garcia-Molina. SIFT - A tool for wide-area information dissemination. In *Proceedings of the 1995 USENIX Technical Conference*, pages 177–186, 1995.
- [20] Rajendra Yavatkar, James Griffioen, and Madhu Sudan. A reliable dissemination protocol for interactive collaborative applications. In *Proceedings of ACM Multimedia*, pages 333–344, San Francisco, CA USA, 1995. ACM.

PART TWO

Experimental Results

On the Integration of the UMTS and B-ISDN system

*G. Karagiannis, B.J.F. van Beijnum, I.G.M.M. Niemegeers
Centre for Telematics and Information Technology (CTIT)
University of Twente, P.O.Box 217, Enschede, the Netherlands
Phone: +31 53 489 3747; Fax: +31 53 489 3247;
Email: karagian@ctit.utwente.nl*

Abstract

The main theme of this paper is the presentation of some critical issues such as transcoding, transport interworking, handover, congestion control and network scalability, that influence the integration of the UMTS and B-ISDN. Solutions to these issues will also be discussed. Furthermore, we will address a number of quantitative (i.e. performance) aspects of an integrated UMTS/B-ISDN system.

Keywords

Mobile communication, B-ISDN, intelligent networks, interworking, transcoding, handover, network scalability, congestion control, performance analysis.

1 INTRODUCTION

The use of mobile communications has increased rapidly over the last fifteen years and a similar growth is being expected for the next ten years. In the early 80s the first generation analogue cellular phones were introduced and the main focus was put on the business sector. The second generation of mobile telecommunication systems has recently been introduced. It is expected that this second generation will be adequate to fulfil the needs until the beginning of the next century. It has been predicted that by the year 2000 the European market will have over one hundred million mobile terminals. By that time, there will be a need for a new generation of mobile telecommunication systems.

To fulfil these future needs, the MONET project (MOBILE NETWORKS) addressed the development of a third generation mobile system which is known as Universal Mobile Telecommunication System (UMTS). The MONET project was sponsored by the European Committee as part of the RACE-II programme.

One of the main objectives of the MONET project was to develop a system in which UMTS and B-ISDN are integrated so that broadband mobile services can be provisioned.

In order to allow such an integration of UMTS and B-ISDN a number of critical technical problems had to be solved. This paper gives an overview of these problems and discusses possible solutions for them.

This paper is organised as follows. In Section 2 we give an overview of the UMTS/B-ISDN architecture. Typical problems related to mobile communication are: transport interworking, transcoding and handover. They are discussed in Sections 3. Two more issues are addressed in Section 4 and 5: scalability and congestion control. Some first experimental performance analysis results are presented Section 6. In Section 7 conclusions are presented.

2 UMTS/B-ISDN ARCHITECTURE

The design and standardisation of the integrated UMTS/B-ISDN system takes place in a stepwise fashion. It starts with user and functional requirements and finally results in an overall architecture: a blueprint of possible realisations of the integrated UMTS/B-ISDN system (MONET, 1995c). The design trajectory followed for UMTS/B-ISDN integration is based on the ITU-T 'stage method' (ITU-T, I.130). An important aspect in the design of the UMTS/B-ISDN architecture is the specification of mobility procedures and their mapping onto system blocks and interfaces.

Each procedure is specified by a single *functional model* (FM). A functional model itself, may be specified in terms of several *functional entities* and their *interactions*. These interactions between functional entities are specified in terms of *information flows* (IFs).

In the MONET project, UMTS has been specified as well as the integration of UMTS and a B-ISDN backbone network. As far as mobile communication is concerned, MONET uses concepts encountered in intelligent networks (IN), for details we refer to (Katoen, 1995), (Broek, 1995), (Karagiannis, 1996) and (MONET, 1995c). In our study we go one step further and consider a backbone network that itself is an integration of B-ISDN and Intelligent Network (IN), the European ACTS project INSIGNIA (Integration of IN and B-ISDN on ATM platforms) is specifying and implementing such an integration (INSIGNIA, 1996).

Figure 1 illustrates the interconnection of the basic physical entities in a UMTS/B-ISDN architecture. Starting at the user side, there are fixed terminals (FT) and mobile terminals (MT). A fixed terminal is directly connected to the backbone B-ISDN network. The backbone network itself may be an integration of B-ISDN and an intelligent network (IN).

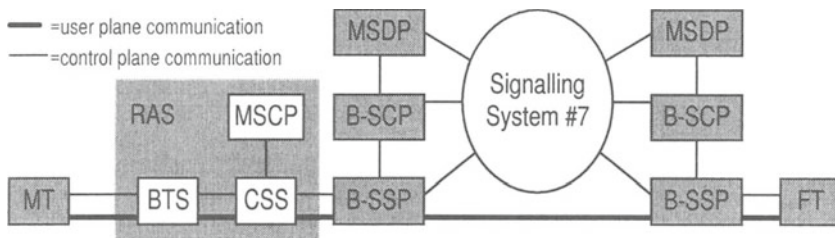


Figure 1 Overview of the integrated UMTS/B-ISDN architecture.

In this case, the backbone network would include physical entities such as: broadband service switching point (B-SSP) that implements UNI/NNI switching and IN interaction; broadband service control point (B-SCP) that implements the service control logic to provide intelligent network services (this includes mobility specific control logic); mobile service control point (MSCP) that implements mobility specific service control logic (such as handover); and mobility service data point (MSDP) that implements a set of functions for mobility specific service data (for instance, mobile terminal location).

A mobile terminal has access to the backbone B-ISDN network via a Radio Access System (RAS), in which the Base Transceiver Station (BTS) manages radio access and the Cell Site Switch (CSS) takes care of call control and it controls the connection to the backbone network.

For the communication between the physical entities in the integrated UMTS/B-ISDN network, a set of protocol stacks have been defined. As far as NNI is concerned we refer to (Colombo, 1994). Our focus will be on the UNI. Figure 2 shows the protocol stacks that are involved in the user plane communication between a mobile terminal and a fixed terminal. UNI radio access incorporates the protocol layers: Radio Physical layer (RPHY), Radio Data Link layer (RDL) and UMTS ATM Adaptation layer (UAL). UNI fixed access incorporated the protocol layers: Physical layer (PHY), ATM and ATM Adaptation Layer (AAL). At application level we have the service application layer (APPL).

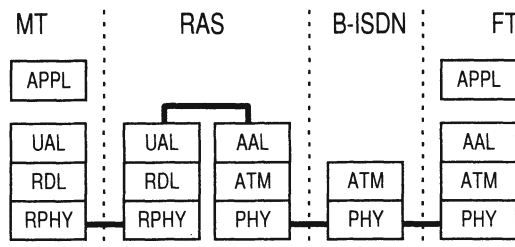


Figure 2 Protocol stacks for user plane communication.

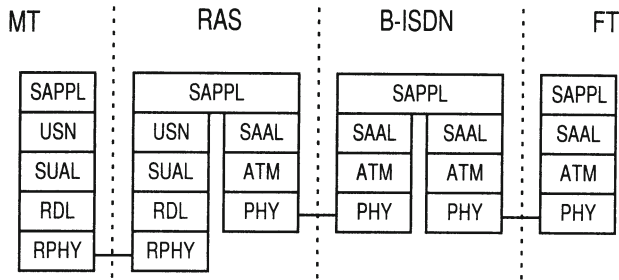


Figure 3 Protocol stacks for control plane communication.

UNI control (or signalling) plane communication is established as shown in Figure 3. Radio access incorporates the layers: Signalling UMTS ATM Adaptation layer (SUAL), UMTS Signalling Network layer (USN) and Signalling Application layer (SAPPL). The UMTS Network layer handles all non-call related mobility functionality such as: paging, location updates (Norp, 1994). Call related signalling is handled by the signalling application (SAPPL). The specification of the Signalling Application layer is one of the results of the MONET project (MONET, 1995c). The fixed part of the UNI control plane needs enhancements of existing UNI signalling in order to support call related mobility functions.

3 INTEGRATION OF UMTS AND B-ISDN

In this section the three major problems encountered in integrating UMTS and B-ISDN, namely transport interworking, transcoding and handover, are discussed.

3.1 Transport interworking

The different UMTS and B-ISDN protocol stacks require interworking functions that extract information from one protocol stack and reformat it for onward transmission via a different protocol stack. Interworking might be defined at different levels in the protocol stack.

In the MONET project three different options for interworking have been identified, for a detailed discussion we refer to (MONET, 1995a), (Marchent, 1996). We give a short explanation of the options and use Figure 2 as reference configuration:

- *Interworking at source (application) level:* This interworking type is service dependent (Immonen, 1995), detailed knowledge and requirements of the data source must be known to control any interworking. The UAL (in the RAS) must interwork with a suitable AAL (in the RAS), this AAL corresponds to the AAL used by the other end user (FT).

- *Interworking at AAL level:* This interworking type is service independent, i.e. it does not need source information to control the interworking. AAL packets are transported transparently to the mobile terminal (MT). The interworking could be achieved in the following way. The MT service application (APPL) encapsulates AAL packets in UAL packets, and the FT service application (APPL) encapsulates UAL packets in AAL packets. At the RAS the interworking from the MT to the FT is performed by extracting the AAL packets from the UAL packets. In the opposite direction the RAS provides interworking by extracting the UAL packets from the AAL packets. The transport interworking function is provided by using the information carried by the AAL headers.
- *Interworking at ATM level:* This interworking type is also service independent (see also (Immonen, 1995)), where ATM cells are transported transparently through the air interface. The RAS provides interworking between the UAL and the ATM layer. Therefore, the AAL layer is not used in the RAS. The interworking could be achieved in the following way. The MT APPL first encapsulates AAL packets into ATM cells, and subsequently the ATM cells are encapsulated in UAL packets. The FT APPL encapsulates UAL packets into AAL packets. At the RAS the interworking from the MT to the FT is performed by extracting the ATM cells from the UAL packets. In the opposite direction the RAS provides interworking by extracting first the AAL packets from the ATM cells and subsequently by extracting the UAL packets. The ATM cell header is used to control the transport interworking function.

Which of the three interworking options can be applied best is still subject to further research, performance evaluation could be one of the important tools to guide this design decision.

3.2 Transcoding

Transcoding is an interworking function that changes the source coding format of the user information transported through several links during an end-to-end connection.

In present-networks such as N-ISDN and GSM (Global System Mobile) transcoding only occurs on network boundaries, because within one network exactly one coding scheme is used for a specific service (i.e. telephony). The advantage of this topology is that the network knows whether transcoding is required before the call is routed to the destination terminal. In this way the network is able to select a route in which a transcoder is permanent physically connected, such that no additional switching capacity is required and no call set-up delay occurs.

With the development of multimedia applications an increase of the number of audio and video coding schemes is foreseen. Therefore, in the future integrated UMTS and B-ISDN system a generic control function for transcoders can be

foreseen that is able to insert transcoders where necessary, in a fashion transparent to the user. The transcoding function is in this case implemented as a Specialised Resource Function (ITU-T, Q.1214).

A disadvantage of such a transcoding scheme, however, is an increased overhead in the network for establishing calls and, correspondingly, an increased call set-up delay time. Therefore, for the most commonly used services there will continue to be a strong need for standardising the coding formats.

It is foreseen that in the integrated UMTS and B-ISDN system the speech transcoding will be the most applied transcoding type. Therefore it could be investigated whether a dedicated “telephony with compressed voice” service can be introduced, see also (ATM Forum, 1994). The main transcoding functions that will be needed for this service type relate to the speech transcoding between ‘compressed voice’ and the standard PCM (Pulse Code Modulation) speech coding format as used in N-ISDN.

In the MONET project (MONET, 1995b) an implementation of the ‘compressed voice telephony’ service is presented. A terminal (mobile or fixed) indicates the compressed voice telephony service and a corresponding cell rate.

If the call is routed to a gateway (similar to the N-ISDN gateway), the gateway applies speech transcoding to 64 kbps PCM and uses signalling messages to the B-ISDN according to the normal narrowband telephony service.

If the call is routed towards a terminal (mobile or fixed), the destination exchange allows the destination terminal to negotiate the type of speech coding (to allow interworking with terminals that do not support compressed voice). To this end the destination exchange adds an information element for 64 kbps PCM encoding to the set-up message and also the bandwidth is negotiated (ITU-T, Q.2962) with a normal peak cell rate of 64 kbps and an alternative peak cell rate according to the compressed voice encoding scheme. If the destination terminal supports the compressed voice service it will select this and no interworking is required. If the destination terminal selects the PCM coding, the destination exchange has to insert a transcoder at connect time.

Further investigations could be accomplished to identify efficient algorithms for the control of the transcoding functions, i.e. identification of the coding types used at the source and destination and of the location of the transcoding function.

3.3 Handover

During a call, handover (HO) is required to transfer the MT connection to each new BTS when the MT roams. From the user's point of view, handover should be seamless. This implies that a handover should not degrade the QoS (Quality of Service) for the connection as perceived by the user, for the support of UMTS/B-ISDN services, such as Broadband Video on Demand and Broadband Video Conference (INSIGNIA, 1996).

In (Karagiannis, 1997) two different HO mechanisms that affect both user and signalling part are described. The first mechanism is the HO synchronised switching mechanism and the second is the HO with multicast support mechanism. The first handover mechanism is relatively easy and it does not use any difficult ATM multicasting nor resynchronisation in the network. The scenario assumes that there is sufficient time available where all the data and history information of the old path can be transferred to the MT before the actual handover to the new path takes place. This means that time is available to empty the buffers on the old path and to stop the interleaving and transcoding process on the old path. It is possible that this time is not available. This implies that the time between a handover decision and the actual handover is too short to end the transmission on the old path gracefully (e.g., ending the interleaving matrix, ending transcoder functions, emptying intermediate buffers). A possible solution for this problem is given by the second HO mechanism where in the downstream direction multicast connections are used in the core network. By using multicast connections to all possible new target RAS systems, the history can already be built up on the possible target paths. At the time the handover needs to take place the history on the old path can be ignored, and the history at the new path can be used to resume transmission. The HO mechanisms require use of end delimiters. When information is switched between old and new paths (both upstream and downstream), the last information on the old path is marked with an end delimiter. The destination will receive information from the old path until this end delimiter and then switch to receiving information from the new path. The end delimiter might be implemented in the ATM header or as an Operations and Maintenance ATM cell.

4 NETWORK SCALABILITY

For any innovative communication system to be commercially interesting it is of crucial importance to be 'scalable'. Network scalability refers to the ability to increase the 'size' of the network while maintaining Quality of Service and network performance criteria. An initial list of distinct, though not necessarily independent, parameters for network size are:

- number of nodes and links in the network;
- number of users connected to the network;
- geographic spread covered by the network, and
- number and type of services provided by the network.

We shall name these parameters the *scalability parameters*. When increasing the size of a network, we want to meet certain network performance criteria. To evaluate the quantitative aspects of a network, hence its performance, we use *network performance measures*. According to (ITU-T, I.350) three main performance criteria can be distinguished, these are: speed, accuracy and

dependability. Specific performance measures related to these criteria are: throughput, error probability and blocking probability.

Basically, the investigation of network scalability is an analysis or evaluation of the network performance measure as function of the scalability parameters. This concept can be brought one step further in which we take so called *resource parameters* into consideration as being our instruments to change and improve the network performance. Within the context of the UMTS/B-ISDN integration a small list of such network resource parameters is shown in Table 1. Based on these three notions (scalability parameters, performance measures and resource parameters) we can construct an iterative procedure for a scalability experiment. It comprises the following steps:

- Step 0:
 - Select network performance measure(s) of interest.
 - Select the scalability parameter(s) of interest and identify their value domain.
 - Identify the desired relationship of the performance measure as function of the scalability parameter.
- Step 1:
 - Determine the performance measure as function of the scalability parameter (e.g. through discrete event simulation).
- Step 2:
 - If the result satisfies the desired relationship then the scalability experiment halts, else the cause(s) of the scalability problem should be identified.
- Step 3:
 - Express the problem in terms of one or more changes in the resource parameters, apply those changes to the network (model).
 - Return to Step 1.

In Figure 4 a scalability experiment is illustrated. When starting such an experiment, we first define the performance measures of interest (for instance throughput), the scalability parameters that we want to consider (for instance number of users) and finally we define the desired value of the performance measure as function of the scalability parameter (assume this is curve Y_0 as shown in Figure 4). In the next step we determine, given the network model, the actual value of the performance measure as function of the scalability parameter resulting in graph Y_1 . Assume that we have identified the scalability problem and that we have found a way to solve the problem, in the next iteration simulation results produces curve Y_2 . By further iteration steps we obtain curve Y_3 , which satisfies the desired performance as function of the scalability parameter. We can therefore halt the scalability experiment.

Table 1 Examples of UMTS/B-ISDN resource parameters

<i>resource parameter</i>	<i>physical entity</i>	<i>explanation</i>	<i>affected performance measure</i>
processing speed	B-SCP, B-SSP	speed at which a message can be processed	delay
routing algorithm	B-SSP	the selected path along which connections are set up	blocking probability
scheduling	RAS, B-SSP, B-SCP	the order in which incoming messages are being processed	delay, blocking probability
service load sharing	B-SCPs	distribution of IN service control over several B-SCPs	delay, blocking probability

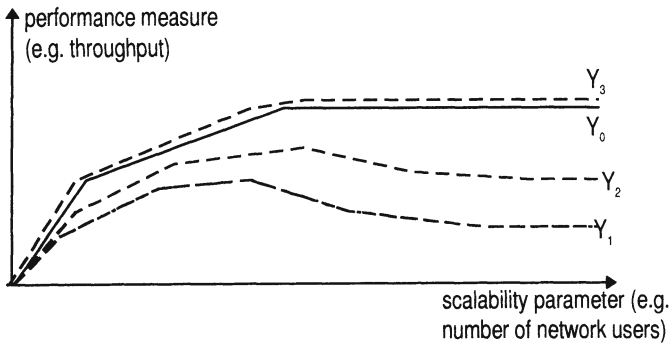


Figure 4 Illustration of a scalability experiment.

5 CONGESTION CONTROL

When the integrated UMTS/B-ISDN architecture has to process an incoming load larger than its capacity, the performance (for example throughput) of the system degrades and eventually the users will experience long delays or a high percentage of service refusals (blocked calls). This state of the system is called congestion. Congestion problems and associated performance degradation can be solved by using appropriate congestion control mechanisms. These mechanisms can be characterised as combinations of admission and flow control algorithms, see also (Schwarz, 1995).

The admission control mechanisms will impose to a receiving node, e.g. MT, to admit a call only if sufficient end-to-end bandwidth is available, i.e. from the calling party to the called party. But in the integrated UMTS and B-ISDN system a mobile terminal can move from a good QoS radio cell to a poor QoS radio cell

and the network node (B-SSP) that is involved in the provisioning of the service must adjust its transmission rate accordingly. This adjustment will create significant problems in the admission control for multimedia applications.

The flow control mechanism is mainly information rate control. The mechanism is activated by a receiving node, e.g. CSS, and throttles the rate at which information is sent by its source, e.g. B-SSP. The flow control mechanism that may be used in the integrated UMTS and B-ISDN system differs from the one used in current fixed networks in the following:

- The variation of the transmission medium characteristics of wireless links (in different radio cells, or even within one radio cell).
- Handling of multiple receivers (mobile and fixed) connected in multiparty call associations.

Because of these differences, it is expected that flow control may be activated on traffic sources more often by mobile receivers (scarce radio resources) than by fixed receivers. An other problem occurs when conflicting flow control requests are issued by several receivers, which may either be fixed or mobile, that are involved in the same call association.

A possible new problem that occurs is due to mobility features, such as handover, which may have a considerable effect on the B-SCP processing. In case the B-SCP has to support both MT and FT in a service session, it is likely that the B-SCP will reach a congestion state sooner, when compared to a situation in which only FT are involved in a service session. When congestion in the B-SCP occurs, and thus the congestion control mechanism is active, the mechanism has to ensure that each source (relative to the B-SCP) gets a, predefined, fair share of the B-SCP processing capacity.

In the literature, we refer to (Smith, 1995), (Pham, 1994), (Bellcore, 1993) and (Turner, 1986), different congestion control mechanisms for fixed systems are described. These could be used as a starting point for the development of new congestion control mechanisms to solve the congestion problems mentioned earlier. In the following, we briefly describe a set of criteria that might be used to compare congestion control mechanisms.

The effectiveness of a congestion control mechanism depends heavily on the system design and on the traffic situation. An effective and efficient method of how a system can deal with moderate traffic overload situation is to adapt system and, only under more extreme traffic overload conditions, to adjust traffic sources to system capacity (Houck, 1994). The main objectives of congestion control can be summarised as follows:

- Provide good throughput even in extreme overload situations.
- The message delays under overload situations should not be much larger than those in normal situations.
- Provide high reliability. Even under high overload situations the system should not break down.

- Ensure system fault tolerance and responsiveness.
- In the situation when service degradation occurs, ensure that this is accomplished gracefully.
- Ensure that under overload situations the system capacity is allocated to the system users on a fair predefined manner.

Here we describe some of the important criteria that can be used to analyse congestion control mechanisms, see also (Smith, 1995), (Korner, 1994), (Hubig, 1994) and (Houck, 1994).

1. *Robustness*: A congestion control mechanism should ensure that the controlled entity (B-SSP or B-SCP) will provide a near maximum throughput even in extreme overload situations. ITU has recommended in (ITU-T, Q.543) that with an overload of 50% above the engineered capacity, the throughput must not fall below 90% of the engineered capacity. The ITU-T recommended throughput curve criterion is a robustness criterion.
2. *Overhead*: The amount of processing capacity needed by the congestion control algorithm should be small relative to the real-time spent for processing input traffic.
3. *Traffic input message delays*: The traffic input message delays during overload should not be much larger than those under normal conditions.
4. *Responsiveness*: This is a superposition of the following times:
 - *detection*: the time between the onset of a congestion event in the system until the time that the control mechanism enters a state denoting that the system is in congestion;
 - *reaction*: The time between the detection of a congestion situation in the system until the time that the control mechanism takes an action to attempt to relieve the congestion;
 - *operation*: The time between the initiation of an action on the part of the congestion mechanism in order to relieve the event and the time that the system resumes normal operation.
5. *Control of the non-essential processing activities during overload*: The non-essential processing activities such as administrative and/or maintenance activities can be suspended during overload situations.
6. *Control of the processing activities that ensure system fault tolerance*: The congestion control mechanism should ensure that the system fault tolerance, e.g., detection and recovery from data errors, under overload situations should not be adversely affected.
7. *Oscillations of congestion states due to statistical fluctuations of traffic*: The congestion control mechanism should tolerate (not be activated during) normal traffic fluctuations. This criterion can be applied to analyse the congestion detection method used in the congestion control mechanism.
8. *Throughput degradation in adjacent nodes*: The congestion control mechanism used in a congested part of a network should not slow or stop the processing activities in a way that would create congestion in adjacent nodes.

9. *Fairness*: A congestion control mechanism is fair if the system capacity (B-SSP or B-SCP node capacity) is allocated to the system users in accordance with some predetermined definition of equity. An effective fairness criterion is quite difficult to specify. Some important factors that can be used to specify the fairness criterion are the following:
 - the rate of reattempts of the rejected system users. This rate should be kept relatively low;
 - the services used. System users that are using business services (e.g. Video Conference) may require and may obtain more system capacity than users that are requiring domestic services (e.g. Video on Demand);
 - the time of the day that a specific service is used. For example a VoD service system user may obtain in the evening hours more system capacity than in the morning hours;
 - frequent system users may also experience a relatively low rejection rate.
10. *Implementation complexity*: An important qualitative criterion for the analysis of congestion control mechanisms is related to the implementation complexity of the mechanism. If the implementation of the mechanism will demand high costs (economical and/or technical) it may not be possible to realise.

6 PERFORMANCE MODELS AND ANALYSIS

To emphasise the need for congestion control and scalability, we analysed the performance of a possible UMTS/B-ISDN configuration, using discrete event simulation techniques.

The configuration that we considered consists of a single B-SCP to which one B-SSP is connected as shown in Figure 5. The B-SSP is connected to one RAS and to n FTs. The RAS is connected to m MTs. A Video Server (VS) is connected to the B-SSP, it contains the digital video information (INSIGNIA, 1996).

The RAS, B-SSP and B-SCP entities are modelled as single queue, single server queueing stations. The B-SSP and B-SCP queues are assumed to be finite (50 messages). The RAS queue has infinite length. The VS and the set of FTs and MTs are modelled as infinite server queueing stations.

The scheduling strategies have been selected as follows: The B-SCP and the RAS use FIFO scheduling; the B-SSP uses FIFO scheduling with priority such that call request in progress are assigned a higher priority than new arriving call request.

We have adopted the B-VoD information flows defined in the INSIGNIA project (INSIGNIA, 1996). Message processing times have been defined on basis of message length, the values are shown in Table 2 (Bernabei, 1996).

With realistic values for the parameter α the B-SCP is the bottleneck in network performance, that is, the B-SCP is the entity that experiences congestion first.

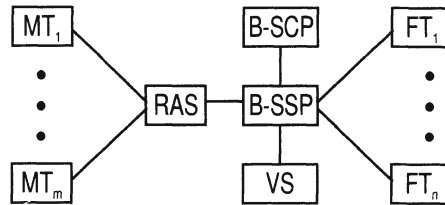


Figure 5 The UMTS/B-ISDN experiment configuration.

Table 2 Performance model parameter values

Physical entity	Message(s)	Processing time (0.1 ms)
B-SSP	AddPartyAndBearerToSession	10
	RequestReportSSMChange	9
	AddBearerToSession	8
	JoinPartToSessionAndLinkLegToBearer	5
	DropParty, PromptAndCollectUserInfo, PlayAnnouncement	3
	ReleaseSession	1
B-SCP	ServiceRequest	14 α
	ReportSSMChange	6 α
	CollectedUserInfo	2 α
	PlayAnnouncement	α
RAS / VS /	Setup	4.5
MT / FT	CallProceeding, Connect	1.5
	ConnectAcknowledge, Release, ReleaseComplete	1

Within the simulation model, traffic is generated using a Poisson arrival process. The traffic generated by the fixed users is 6.6 times higher than the traffic generated by the mobile users (MAGIC, 1995). In the simulation experiments, the traffic generated by the mobility procedures such as paging, location updates and handovers have not been included, but will be in further experiments. In the analysis we have focused on network throughput as function of the call request rate.

The scalability experiments that we have performed are based on the procedure described in Section 4 of this paper. We assume the 'desired' throughput as shown in Figure 6. The resource parameter considered in the experiment is the B-SCP processing speed (which is indirect proportional to parameter α in Table 2), and as scalability parameter we have selected the number users (which alternatively is reflected in the B-VoD call requests rate).

Three successive simulation experiments have been conducted with $\alpha=6$, $\alpha=2.7$ and $\alpha=1.67$ respectively, the point estimates are shown in Figure 6.

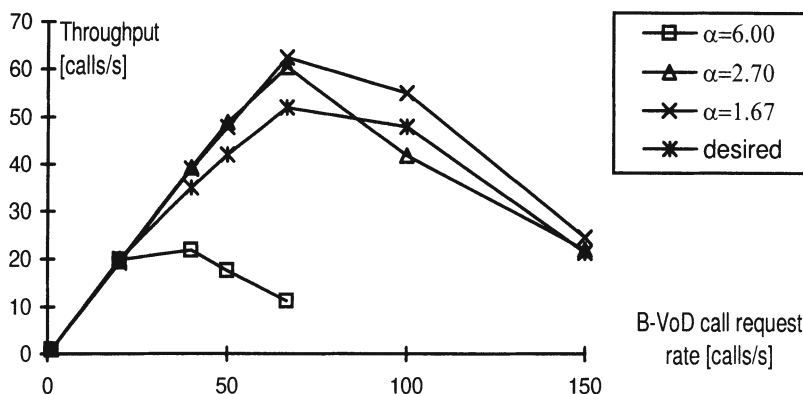


Figure 6 The network throughput as function of the call request rate.

In the analysis the confidence coefficient has been set to 0.99, the resulting confidence interval at each point is (in the worst case) $\pm 13\%$ of the point estimate. We can conclude that the ‘desired’ throughput is satisfied for $\alpha = 1.67$.

7 CONCLUSIONS

This paper describes some of the critical issues that affect the integration of the UMTS and B-ISDN systems and some solutions are proposed. Furthermore, it addresses a few quantitative (i.e. performance) aspects of an integrated UMTS/B-ISDN system.

The different UMTS and B-ISDN protocol stacks require interworking functions that extract information from one protocol stack and reformat it for onward transmission via a different protocol stack. Interworking might be defined at different levels in the protocol stack.

Transcoding is an interworking function that changes the source coding format of the user information transported through several links during an end-to-end connection.

During a call, handover is required to transfer the MT connection to each new BTS when the MT roams. From the user's point of view handover should be seamless which implies that a handover should not degrade the QoS (Quality of Service) for the connection as perceived by the user for the support of UMTS/B-ISDN services, such as B-VoD and B-VC.

For any innovative communication system, scalability of the network is of crucial importance for keeping it commercially interesting. Scalability is the ability to increase the size of a network while maintaining the Quality of Service. The size of a network may refer to several different things, such as: the number of users, the number of network nodes and the geographical spread of the network.

When the integrated UMTS/B-ISDN architecture has to process an incoming load larger than its capacity then the performance (for example throughput) of the system degrades and eventually the users will experience long delays or a high percentage of service refusals (blocked calls). This state of the system is called congestion. The congestion state can be changed (resolved) by using appropriate congestion mechanisms.

To emphasise the need for congestion control and scalability experiments we investigated the performance of a possible UMTS/B-ISDN configuration by performing simulation experiments.

8 ACKNOWLEDGEMENTS

This paper is based on work in the RACE II project MONET (Mobile Networks) and on the ACTS project INSIGNIA (IN and B-ISDN Signalling Integration on ATM platforms) in which the authors participated. The views expressed in this paper are those of the authors and not necessarily those of the projects as a whole. The members of the MONET project, in particular Toon Norp and Marc de Lignie are acknowledged for their help and discussion.

The authors would also like to express their gratitude to Victor Nicola and Phil Chimento for discussions on the subject, and to the European Committee for initiating and sponsoring the RACE and ACTS programmes.

9 REFERENCES

- ATM Forum (1994) User-network interface (UNI) specification, version 3.1.
- Bellcore (1993) AIN SCP generic requirements, GR-1280-CORE, Bellcore, Issue 1. August.
- Bernabei, F., Chierchia, G., Gratte, L., and Cuomo, F (1996) Messages weight assignment, internal ACTS INSIGNIA communication.
- Broek, J. van der and Katoen, J.-P. (1995) Reference configurations for UMTS, in *Proceedings of the 2nd International Workshop on Mobile Multi-Media Communications*, A1/1/1-6.
- Colombo, G. and Hegeman, H. (1994) Network architecture and functionalities for UMTS, in *Proceedings of Wireless Communication Networks* (eds. J. Weber, J. Arnbak and R. Prasad), Volume 3, 844-851.
- Houck, D.J., Meier-Hellstern, K.S., Saheban, F. and Skoog, R.A. (1994) Failure and congestion propagation through signalling controls, in *Proceeding of the 14th International Teletraffic Conference* (eds. J. Labetoule and J.W. Roberts), 367-376.

- Hubig, W., and Weber, D. (1994) Overload control in ISDN PABXs, in *Proceeding of the 14th International Teletraffic* (eds. J. Labetoule and J.W. Roberts), 243-252.
- INSIGNIA (1996) First trial: description of the selected services, project document reference number AC068/CLT/111/DS/P/001/b1 (public).
- ITU-T I.130, ITU-T Recommendation I.130, .Method for the characterisation of telecommunication services supported by an ISDN and network capabilities of an ISDN.
- ITU-T I.350, ITU-T Recommendation I.350, General aspects of service and network performance in digital networks.
- ITU-T Q.1214, ITU-T Recommendation Q.1214, Distributed functional plane for Intelligent Network CS-1.
- ITU-T Q.2931, ITU-T Q.2931 reference, DSS2 access signalling for B-ISDN.
- ITU-T Q.2962, ITU-T Q.2962 reference, DSS2 bandwidth negotiation capability.
- ITU-T Q.543, ITU-T recommendation Q.543, Blue Book, Digital exchange performance design.
- Karagiannis, G., Katoen, J.-P. and Niemegeers, I.G.M.M (1996) B-ISDN to the cell site switch versus B-ISDN to the mobile terminal, in *Proceeding of the IEEE ICCS/ISPACS '96*, Singapore, Volume 2, 629-633.
- Karagiannis, G., Lignie, M.C. de, Bie, J. de and Niemegeers, I.G.M.M (1997) Handover mechanisms in ATM-based mobile systems, to be published in *Journal of Selected Areas in Communications*.
- Katoen, J.-P. (1995) Functional Integration of UMTS and B-ISDN, in *Proceedings of the 45th IEEE Vehicular Technology Conference*, Chicago (Illinois), Volume 1, 160-164.
- Korner, U., Nyberg, C., and Wallstrom, B. (1994) The impact of new services and new control architectures on overload control, in *Proceeding of the 14th International Teletraffic Conference* (eds. J. Labetoule and J.W. Roberts), 275-282.
- MAGIC (1995) B-ISDN Signalling performance: issues and analysis, document reference R2044/BTL/DP/DS/P/014/b1 (public).
- MONET (1995a) Interoperability and integration of UMTS in a B-ISDN backbone, document reference R2066/RMR/UNA2/DS/P/100/b1 (public).
- MONET (1995b) Recommendations of UMTS integration scenario's in the B-ISDN backbone, document reference R2066/RMR/UNA2/DS/P/107/b1 (public).
- MONET (1995c) UMTS system structure document, document reference R2066/BT/PM2/DS/P/113/b1 (public).
- Norp, T. (1994) Protocol stacks for an integrated UMTS user access network, in *Proceeding of the RACE mobile telecommunications workshop*, Amsterdam, 310-313.
- Pham, X.H. and Betts, R. (1994) Congestion control for intelligent networks, *Computer networks and ISDN systems*, **26**, 511-524.

- Schwartz, M. (1995) Network management and control issues in multimedia wireless networks. *IEEE Personal Communications Journal*, **2**(3), 8-16.
- Smith, D.E. (1995) Ensuring robust call throughput and fairness for SCP overload controls. *IEEE/ACM Transactions on networking*, **3**(5), 538-548.
- Turner, J.S. (1986) New directions in communications. *IEEE Communication Magazine*, **25**(10), 8-15.

10 BIOGRAPHY

G. Karagiannis received a bachelors degree in Electrical Engineering at the T.E.I Athens, Greece in 1987. In 1988 he joined the laboratory of Network Theory of the Electrical Engineering department at the University of Twente, the Netherlands, where he was involved in VLSI design projects. He received a MSc. degree in electrical engineering at the University of Twente in 1993. In 1995 he graduated for a post graduate designers course at the Tele Informatics and Open Systems (TIOS) group at the University of Twente.

In 1995 he joined the CTIT at the University of Twente. His areas of interest include mobile communication systems, network control mechanisms, performance modelling and analysis. He has actively participated in the RACE-II project MONET and is presently actively involved in the ACTS project INSIGNIA.

B.J.F. van Beijnum was born in Nijmegen, the Netherlands in 1961. In 1985 he received a MSc. degree in electrical engineering and in 1995 he received a Ph.D. degree at the University of Twente. In 1985 he joined the laboratory for network theory of the electrical engineering department at the University of Twente where he worked on formal description and verification methods and their application for VLSI design. In 1994 he joined the Inter University Micro Electronics Centre (IMEC) in Leuven, Belgium.

Since 1995 he is a CTIT staff member at the University of Twente. His present areas of interest include high speed communication network architectures, signalling, performance analysis and performance measurements. He is actively involved in the ACTS project INSIGNIA.

I.G.M.M. Niemegeers was born in Gent, Belgium in 1947. He received a degree in electrical engineering from the Rijksuniversiteit Gent in 1970. In 1972 he received a MSc. E. degree in computer engineering and in 1978 a Ph.D. degree from Purdue University in West Lafayette, Indiana, USA. From 1978 to 1981 he was a designer of packet switching networks at Bell Telephone Mfg. Cy., Antwerp, Belgium.

Since 1981 he is a professor at the Computer Science Department of the University of Twente, Enschede, the Netherlands. He is presently Scientific

Director of the Centre for Telematics and Information Technology of the University of Twente. His areas of interest are communication systems and performance analysis. He is active in research on integrated networking, high-speed networking, B-ISDN, optical networking, performance analysis and performability.

Measuring the Behavior of a World-Wide Web Server

Jussara M. Almeida
(jussara@dcc.ufmg.br)

Virgilio Almeida^{*1}
(virgilio@bu.edu)

David J. Yates^{*2}
(djy@bu.edu)

Depto. de Ciência da Computação,
Universidade Federal de Minas Gerais,
Belo Horizonte,
Minas Gerais 31270-010, Brazil

Computer Science
Department,
Boston University,
Boston, MA 02215, USA

Abstract

Server performance has become a crucial issue for improving the overall performance of the World-Wide Web. This paper describes Webmonitor, a tool for evaluating and understanding server performance, and presents new results for a realistic workload.

Webmonitor measures activity and resource consumption, both within the kernel and in HTTP processes running in user space. Webmonitor is implemented using an efficient combination of sampling and event-driven techniques that exhibit low overhead. Our initial implementation is for the Apache World-Wide Web server running on the Linux operating system. We demonstrate the utility of Webmonitor by measuring and understanding the performance of a Pentium-based PC acting as a dedicated WWW server. Our workload uses a file size distribution with a heavy tail. This captures the fact that Web servers must concurrently handle some requests for large audio and video files, and a large number of requests for small documents, containing text or images.

Our results show that in a Web server saturated by client requests, up to 90% of the time spent handling HTTP requests is spent in the kernel. Furthermore, keeping connections open, as required by TCP, causes a factor of 2 increase in the elapsed time required to service an HTTP request. Data gathered from Webmonitor provide insight into the causes of this performance penalty. These results emphasize the important role of operating system and network protocol implementation in determining Web server performance.

1 INTRODUCTION

The quality of networked services like the World-Wide Web (WWW) depends on many factors, including performance, reliability, and security. The overall performance of the Web depends on the performance of its main components; namely clients, the network, and servers. The explosive growth of the Web is placing a heavy demand on servers [Birman *et al*, 1996]. As a result, users see slow response times on the most popular sites, which are overrun by millions of requests per day. Thus, server performance has become a critical issue for improving the quality of service

^{*1} On sabbatical at Boston University from Universidade Federal de Minas Gerais. Partially supported by CNPq-Brazil.

^{*2} This work was supported in part by NSF grants CDA-9529403 and CDA-9623865.

on the World-Wide Web. In order to improve Web server performance, we need to understand how server behavior differs in response to different types of requests, such as requests for small HTML documents, or for large audio and video files. We need to gain insight into server behavior under heavy load in the presence of such heterogeneous requests. In particular, we need to assess the impact of operating system and network protocol implementation on server performance. This suggests the need for quantitative measurements that show how system resources are being utilized when servicing HTTP requests.

Despite the importance of measuring and understanding the behavior of Web servers, there are no freely available performance tools that give detailed information about server behavior. In this paper, we describe and present results from a prototype tool (called *Webmonitor*) that does just this. For an HTTP workload, Webmonitor measures activity and resource consumption, both within the kernel and in HTTP processes running in user space. Webmonitor is implemented using an efficient combination of sampling and event-driven techniques that have low overhead (less than 4%), and therefore does not significantly perturb server behavior. Our initial implementation is for the Apache WWW server running on the Linux operating system.

We demonstrate the utility of Webmonitor by measuring and understanding the performance of a Pentium-based PC acting as a dedicated WWW server. We present results for a workload generated by WebStone [Trent *et al.*, 1995], which is a configurable tool for benchmarking Web server performance, available from Silicon Graphics. We parameterized the server workload generated by WebStone to capture the heterogeneous nature of HTTP requests, using values from [Arlitt *et al.*, 1996]. Specifically, we used a file size distribution with a heavy tail to capture the fact that Web servers must concurrently handle some requests for huge multimedia files and a large number of requests for small HTML and image documents. Such distributions occur in the size of files available at servers, and in files requested by clients [Arlitt *et al.*, 1996, Crovella *et al.*, 1996]. This heterogeneity in workload stresses the limits of the underlying operating system much further than traditional applications [McGrath *et al.*, 1995]. One other important characteristic of our workload (and experiments) is that we do not reuse TCP connections for multiple HTTP requests, as described in [Mogul, 1995a] and the Apache documentation [Robinson *et al.*, 1995]. Thus, we open a new TCP connection for every request. We therefore capture the costs of servicing our workload under the “worst case” assumption of being unable to use persistent connections.

We present two new results from data collected using Webmonitor. First, in a Web server saturated by client requests, we find that up to 90% of the time spent handling HTTP requests is spent in the kernel. Second, that keeping TCP connections open causes a factor of 2 increase in the elapsed time required to service an HTTP request. It is necessary to keep TCP connections open (in the `TIME_WAIT` state) at the server to guard against old data being received by a new connection. Although such problems with the way TCP interacts with HTTP have been pointed out by others [Mogul, 1995c, Mogul, 1995a, Padmanabhan, 1994], we isolate and quantify their impact. Specifically, we show that these lingering TCP connections cause a 33% performance penalty in terms of throughput.

The rest of the paper is structured as follows. Section 2 outlines specific characteristics of the Web that influenced the approach we adopted to measure server behavior. In section 3 we describe the experimental environment that was instrumented

and measured, and the workload used to drive our experiments. Section 4 presents an overview of the Webmonitor architecture and important aspects of its implementation. Next, we present and analyze measurements collected by Webmonitor. We then use the tool to measure the behavior of a busy Web server, and discuss the impact of the Web server implementation on performance. Finally, section 5 summarizes the paper.

2 MEASURING A WEB SERVER

The standard performance tools provided by Unix operating systems include *ps*, *vmstat* and *netstat*. In Linux, all of them collect information from */proc* filesystem [Welsh, 1994]. Although these tools can provide insight into server behavior, they reflect the performance only from a system-wide standpoint. Furthermore, those standard tools may introduce unbearable overhead during the monitoring of a busy Web server [Cockcroft, 1996].

In order to obtain in-depth information about the server behavior, we also need to collect data at the HTTP server level. HTTP servers usually log per-request information in log files, but that is not enough to gain insight in the way system resources are used to service an HTTP workload. Thus, we decided to build a specific tool to monitor the behavior of Web servers and to measure resource usage. In this section, we describe the guidelines and principles we followed to design a Web server performance monitor.

2.1 Characteristics of Web Servers

As pointed out in [Arlitt *et al.*, 1996, Crovella *et. al.*, 1996, Mogul, 1995a, Mogul, 1995b, Almeida Bestavros *et al.*, 1996], there are several characteristics that distinguish Web servers from traditional distributed systems. The following two characteristics have a profound impact on the behavior of Web servers.

(a) Heavy Tailed Distributions

Recent studies [Arlitt *et al.*, 1996, Crovella *et. al.*, 1996] have shown that file sizes in the World-Wide Web exhibit heavy tails, including files stored on servers, files requested by clients and transmitted over the network. A heavy-tailed distribution (e.g., Pareto) is given by $P[X > x] \sim x^{-\alpha}$, as $x \rightarrow \infty$ and $0 < \alpha < 2$. Theoretical heavy-tailed distributions have infinite variance, which, in practical terms, means that very large observations are possible with non-negligible probability. In [Crovella *et al.*, 1996], the authors surveyed a number of WWW servers in the Internet and found evidence of heavy-tailed distributions of sizes of files on the servers. One possible explanation is the presence of large multimedia files that contribute to increase the tail of file size distribution.

(b) Short-lived Processes

Most HTTP server implementations use a new TCP connection for almost every request. Several references [Arlitt *et al.*, 1996, Crovella *et al.*, 1996] report that over 90% of client requests are for small HTML or image files. The combination of these facts explains a common phenomenon that has been observed during the operation

of busy Web servers: the creation of a large number of short-lived processes [Mogul, 1995a, Mogul, 1995b]. This brings new challenges to some operating systems that are not tuned for handling a large number of short-lived processes. Short-lived processes also represent new problems for performance monitoring. Although UNIX provides accurate measurements for processor usage by processes of moderately long duration, the authors in [Somin *et al.*, 1996] point out the problems in trying to measure CPU time used by short-lived individual processes.

2.2 Measurement Approach

The fundamental characteristics of a good measurement tool are low overhead, low interference in the system being measured, and high accuracy. We address these characteristics in the design and implementation of Webmonitor.

Although monitors can provide a great deal of useful data, there are problems with the use of their data for performance modeling. Thus, Webmonitor was designed to provide data for analytical models also. The basic input data required by queueing network models are service demands of a request at a server [Menasce *et al.*, 1994]. Those demands specify the total amount of service time required by a request during its execution at each major component of the server. It is worth mentioning that service demand refers only to the time a request spends actually receiving service. It does not include waiting times. Webmonitor was designed to provide this information, which can then be used to derive the basic data required by analytical and simulation queueing models.

In this section we show the features of Webmonitor that take advantage of World-Wide Web workload characteristics to achieve low overhead and high accuracy.

(a) Monitoring Techniques

Webmonitor uses a combination of sampling and event-driven techniques to collect different levels of information about the operation of a Web server. Sampling-based measurement is used to read counters that are maintained by the kernel. Those counters provide system-level information (e.g., resource utilization, interrupt rates, etc.) as well as network statistics. Because events occur within different modules in a Web server, our monitor supports the concept of different sampling intervals, that are adjusted to the nature of the information being monitored. However, the choice of sampling intervals always represents a trade-off between accuracy and overhead. To do sampling in an efficient way, we made some modifications to the Linux kernel, because of the high volume of requests that would imply high overhead.

Sampling is an inadequate technique to trace the execution of every HTTP request in user space. Thus, for monitoring the execution of every request in the HTTP processes, Webmonitor uses an event-driven technique that required the instrumentation of the server. However, it is important to note that this instrumentation was designed to be lightweight to minimize its effect on server processing.

(b) Classes of Requests

Although it would be desirable to have detailed execution information about each individual request, it is unfeasible, in terms of overhead, to keep track and record this quantity of information. This is especially true for busy Web servers that are

overloaded by requests. A possible solution would be to simply accumulate the execution information for all requests and to calculate average values for the measurements. However, as we saw earlier in this paper, requests for documents at Web servers follow heavy tailed probability distributions, that have very large variance. Thus, average results for the whole population of requests would have no statistical meaning.

As a compromise to keep overhead as low as possible without impairing the accuracy and significance of the measurements, we categorized requests into a small number of classes. A class is defined by a range of file sizes, and these ranges are chosen to reflect a heavy tailed distribution of file sizes on the server. Thus, each class comprises requests that are similar with respect to the size of the files they retrieve. As a result, we group together requests of similar behavior in terms of resource usage, which helps reduce the variance of the collected data.

3 EXPERIMENTAL SETUP

This section explains in detail the WWW server which we used in our experiments. We describe the workload, hardware, and software used to perform the measurements and collect the performance data.

The operating system used is Linux version 2.0.0, which is distributed under the terms of GNU General Public License [Welsh, 1994]. The server software is Apache, version 1.1.1, a public domain HTTP server [Robinson *et al.*, 1995].

Apache was originally based on code and ideas found in NCSA HTTP server [McGrath *et al.*, 1995]. It is “**A PAtCHy** server”, in the sense that it was based on some existing code and a series of “patch files”. Apache can run in two different modes: from the `inetd` system process or, in standalone mode. The main disadvantage of running an HTTP server from `inetd` is that, for each HTTP connection received, a new copy of the server is started from scratch; after the connection is complete, this program exits. Thus, there is a high per-connection overhead. Standalone is therefore the most common mode of operation, since it is far more efficient. The server is started once, and services all subsequent connections.

Another interesting point worth mentioning concerns the management of the HTTP processes. Apache maintains a pool of child server processes to handle incoming requests. On startup, a master server process spawns a pre-defined number of child processes and as the load in the server increases, new processes are spawned and included in this pool. The master process periodically checks the number of idle child processes and dynamically adapts this number to the load it sees. There are pre-defined limits (lower and upper bounds) to the number of idle processes. Besides this, there are also upper bounds for the number of requests each child is allowed to process before it dies and on the total number of child processes running, that is, a limit on the number of clients that can simultaneously connect to the server.

Our Apache server was configured to run in standalone mode. The number of KeepAlive requests per connection [Robinson *et al.*, 1995] was set to 0 (only one HTTP request was serviced per connection). The lower and upper bounds in the number of idle processes were set to 5 and 10, respectively; and the number of requests a child process serves before dying was set to 30. Our hardware platform was an Intel Pentium 75MHz system, with 16 Megabytes of main memory and a 0.5 Gi-

gabyte disk. It has a standard 10 Megabit/second Ethernet card. Linux was installed on the disk on a partition of 416 Megabytes, and a partition of 36 Megabytes was allocated for swap space.

To generate a representative WWW workload, we used WebStone [Trent *et al.*, 1995] (version 2.0), which is an industry-standard benchmark for generating HTTP requests. WebStone is a configurable client-server benchmark for HTTP servers, that uses workload parameters and client processes to generate Web requests. This allows a server to be evaluated in a number of different ways. It makes a number of HTTP GET requests for specific pages on a Web server and measures the server performance, from a client standpoint.

WebStone is a distributed, multi-process benchmark, where a master process spawns, local or remotely, a pre-defined number of client processes. Each client process generates requests to the server and collects the performance statistics. After all clients finish running, the master process collects the client's statistics and calculates the overall server performance during the execution of the workload. In our experiments, the client processes were spread over three machines: two SparcStation 20 (128 and 256 megabytes of main memory and operating systems SunOS 4.1.4 and 5.5) and one SparcStation Ultra (128 megabytes of main memory and SunOS 5.5) In order to generate load for a WWW server, client processes successively request files from the server, as fast as the server can answer the requests. A new request is sent out to the server right after a client receives the answer from a previous request. The main performance measures collected by WebStone are latency and throughput. The former represents the response time to complete a request, viewed from the client side. Throughput is measured in connections per second and also in bytes transferred per second.

Table 1 Characteristics of HTTP Workload

Item	Number of files	File size (KBytes)		Access probability	
		Total	Average	Total	Average
HTML	24	180	7.5	0.192	0.008
Images	29	385	13.28	0.754	0.026
Sound	20	3580	179	0.05	0.0025
Video	4	9216	2304	0.004	0.001

The WebStone workload is defined by the number of client processes and by the configuration file that specifies the number of files, their size and access probabilities. Table 1 gives baseline information for the HTTP workload used in our experiments. The parameters that define the workload are representative of the kinds of workload typically found in busy WWW servers [Arlitt *et al.*, 1996]. It is worth noting that the set of files in this workload consumes 82% of physical memory. Furthermore, once the kernel and HTTP processes are also present in memory, we observe significant disk activity in our experiments.

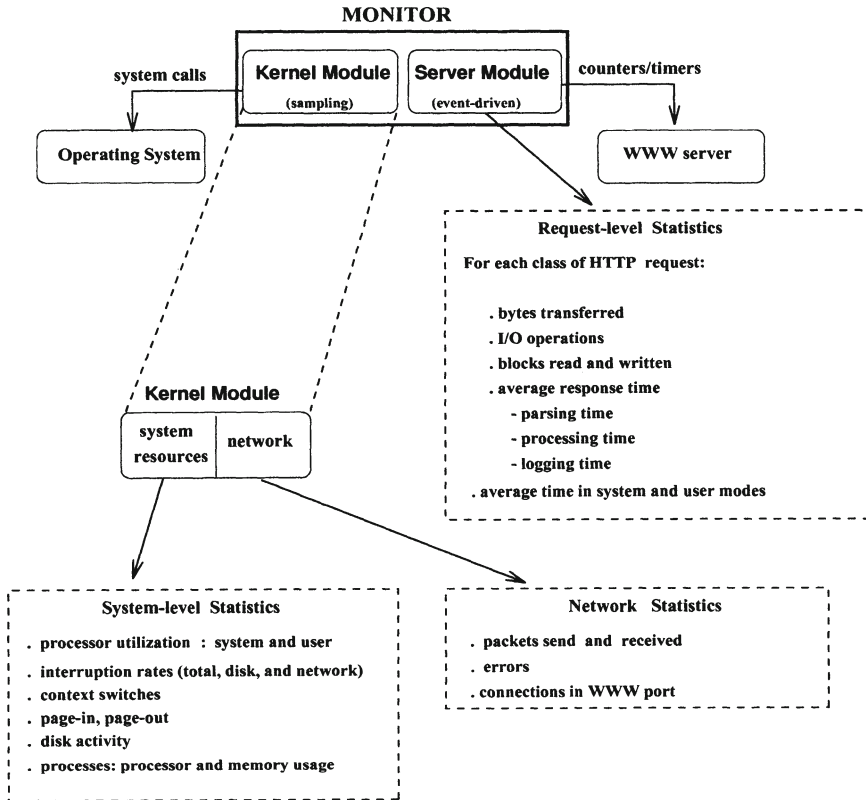


Figure 1 Overview of the Webmonitor

4 ARCHITECTURE OF THE MONITOR

Figure 1 depicts an overview of Webmonitor. The monitor can be seen as a combination of two main components that operate at different levels of the system and collect performance data using different techniques. This division is based on the interaction between the monitor and system, the technique of instrumentation used and the nature of the data collected. The Kernel Module runs independently of the Web server and collects information about the operating system as a whole. The code of the Server Module is actually linked with the server code, and therefore runs as part of the server. It collects information about server performance during the handling of HTTP requests.

4.1 The Kernel Module (KM)

The Kernel Module (KM) collects resource usage data, not only from a system-wide standpoint but also from the Web server viewpoint. The information collected is: processor utilization, disk activity, paging activity, and interrupt rates. This module also collects information about network activity, which is divided into two groups. The first one refers to statistics on communication activities through the Ethernet in-

terface, such as the number of packets transmitted or received, number of errors that occurred during transmission or reception. The second group provides information about the number and state of TCP connections to the HTTP port in the server. The TCP state information is useful for understanding the “lifetime” of connections in the server.

In addition to the three types of system-wide information activity described above, KM also obtains information about certain processes. The information is basically CPU and memory utilization. It also collects the total number of copies of each monitored program (**started processes**) and the number of copies waiting for run time (**running processes**). In our experiments, we chose to monitor the HTTP processes and the kernel processes responsible for swapping and buffer cache management. However, since our results show that the vast majority of system resources are consumed by the HTTP processes, we only present results for these processes.

Usually the Linux kernel keeps performance data internally. They can be read by user programs through the `/proc` filesystem [Welsh 1994]. This is a “virtual file system”, in the sense that its contents are not located on disk but in memory. A read of any file below `/proc` causes data in the kernel to be copied to memory in user space. This information is actually copied as a sequence of ASCII characters. Thus, to find specific data, it is necessary to parse a string for a specific keyword and then read one or more numeric values.

There is one important disadvantage to using `/proc` to gather kernel activity information. If one needs to gather information scattered throughout several kernel data structures, one must perform multiple reads (each of which is a system call), or read very large blocks of data out of the kernel. Both of these alternatives are very expensive. This overhead of reading `/proc`, to get specific but scattered information, is the main reason we decided to implement the KM using four new system calls.

The information gathered by the KM is collected through four system calls that summarize and return specific information about kernel activity in a single buffer. The KM system calls are as follows:

- **my_get_kstats**: returns information about processor utilization, disk activity, paging activity and interrupt rates.
- **my_get_procstats**: returns cpu and memory utilizations for each process with a given command name.
- **my_get_netstats**: returns the number of packets transmitted and received and the number of errors occurred in the network interface.
- **my_get_connstats**: returns the number of connections in each TCP state connected to a given TCP port.

The KM runs as a group of two to four processes, periodically collecting information through the system calls described above. The number of samples, the TCP port to be monitored, the number of different programs to be monitored, the name of them and the number of KM processes spawned are parameters specified in a configuration file [Almeida Almeida *et al.*, 1996].

4.2 The Server Module (SM)

The Server Module (SM) is responsible for collecting information about server performance during the handling of HTTP requests. It is implemented as a library of

routines compiled and linked with the server code. Calls to specific routines were inserted at appropriate points in the server code. Instead of being based on sampling, like the KM described in the last section, the SM collects information based on a trace of events that occur during the handling of a single request. The data collected are: bytes transmitted, connections established, read and write operations, and number of blocks read and written during the handling of the request. Another important piece of information is the processing time at the server to handle a request. The time measured by the SM begins with the establishment of a connection and ends when the server (i.e., HTTP process) is ready to handle the next request. It is broken into three components, which are measured in processor time and in elapsed time. *Parsing time* is the interval that begins just after the establishment of the connection and ends when the header of the request has been parsed and is ready to be processed. *Processing time* is the time spent actually processing the request. It does not include the server logging time. It accounts for the time spent reading the URL (Uniform Resource Locator) and the time needed to move the file from memory or disk to the network. *Logging time* is the time spent performing standard HTTP logging. After logging, a server process is ready to handle a new request.

Unfortunately, the Linux timing routines are not accurate enough to account for the three components of the execution time of a short request. The timing resolution is on the order of 10 milliseconds [Welsh, 1994]. In order to measure parsing, processing, and logging times with greater accuracy, we implemented a “stopwatch” scheme using the `gettimeofday` routine, that returns the elapsed seconds and microseconds since a predefined date. This resolution is because `gettimeofday` reads the time directly from the hardware timer. In order to be timed using a stopwatch, a process must call a system routine to include itself in a *CPU Monitored Processes Table*, located in kernel memory. This routine returns the entry allocated in the table for that process. There are also system calls to *Start* and *Stop* the time accounting. To discount the time that the CPU was used by processes other than the one being monitored, an entry of the *CPU Monitored Processes Table* also contains the time between *Start* and *Stop* spent servicing other processes [Almeida Almeida *et al.*, 1996].

A similar scheme was implemented in order to collect per-process disk activity information. It creates a *Disk Monitored Processes Table*, where appropriate information is kept. To be monitored, a process must allocate an entry in this table through a system call. Every time a disk request from a process being monitored is served, the number of read or write operations and the blocks transferred are registered in its entry in this table.

Each server process collects the statistics described above for the requests that it services. In addition, the SM incorporates the concept of *request classes*. Each request is categorized into one of several predefined classes depending on the size of the file requested. The classes are defined in a configuration file specifying the maximum file size for each class. The statistics collected by a server process are separated by class. Thus, while handling a request, a server updates the counters associated with the class of the request being serviced. In this manner, the SM generates cumulative information for each class and each HTTP server process. To keep overhead low, this information is written to disk by the server processes after 10 requests have been served. After data collection is complete, these cumulative values can be processed to generate other statistics such as averages, variances, etc.

4.3 Monitor Overhead

One of the main concerns in the design of the WWW server monitor was to keep overhead as low as possible. The response time (in seconds) and throughput (conn/s and Mbits/s) measured by WebStone for the server with the monitor (KM and SM modules) were 1.74, 17.35 and 3.91, respectively. Without the monitor, WebStone measured 1.69, 17.96 and 4.02, for the same workload. Thus, the overhead introduced by the monitor is less than 4% for all three measures. We also compared the cost of using our system calls against the cost of obtaining the same information through the `/proc` filesystem. Compared to Webmonitor system calls, collecting the same data via `/proc` is between 5 and 200 times more expensive [Almeida Almeida *et al.*, 1996].

5 RESULTS

Recall that one of the main design goals of our WWW server performance monitor is to understand how time is spent servicing HTTP requests, and how different components of the server software are utilized. The KM addresses this goal by measuring the CPU user and system time, and the rate at which different kernel services are invoked (e.g., read calls per second). The SM addresses this goal by measuring the CPU utilization and latency of servicing requests, as well as tracking per-connection use of some kernel services (e.g., read calls per connection).

We demonstrate the utility of our WWW server performance monitor at the most interesting operating point of the server – when it has just become saturated. To determine the saturation point, we ran experiments varying the number of WebStone clients that communicate with the server. Our results, average values from 3 experiments, are for 30 clients, which cause both the CPU and memory of the server to be utilized at levels greater than 90%. We discuss these results for the server module first, then describe results from the kernel module, and then tie them together. Finally, we present results for experiments where we change the Linux TCP implementation to *not* keep connections open at the server. Comparing these results with our original results shows the effect that keeping TCP connections open has on server performance.

5.1 Server Module Results

Table 2 shows server module (SM) measurements for the three different classes of requests. Recall that these request classes correspond to different file sizes that span a heavy-tailed distribution. Furthermore, each class is representative of an object “class,” as in Table 1. Class 1 requests (for HTML and image documents) are for small files; they have a mean size of 12.1 KB and make up the vast majority of the requests (i.e., 94.6%). Class 2 requests (for audio files) are moderate in size and amount to 5% of requests. Class 3 requests (for video clips) are large (2.3 MB on average) and make up only 0.4% of the workload. The most interesting result in Table 2 lies in the last six rows, which show the processor time and the elapsed time of the three different phases of execution of an HTTP request. These rows show that in most cases the majority of the time spent servicing an HTTP request is spent moving

Table 2 Server Module Results for 30 Clients

	Class 1	Class 2	Class 3
conn/s	16.40	0.88	0.07
Mbits/s	1.55	1.17	1.19
reads/conn	0.03	2.67	34.67
CPU parsetime(ms)/conn	4.78	4.81	3.81
CPU processingtime(ms)/conn	18.75	150.56	2231.35
CPU logtime(ms)/conn	5.28	6.99	9.10
elapsed parsetime(ms)/conn	23.45	21.80	6.20
elapsed processingtime(ms)/conn	155.81	3789.95	60578.90
elapsed logtime(ms)/conn	774.31	940.35	827.42

the requested URL from the filesystem to the network (i.e., processing the parsed request). This is true of CPU time for all three request classes in our workload. Furthermore, the elapsed processing time also dominates the elapsed parse and logging times for moderate and large (Class 2 and 3) requests. The CPU time and elapsed time for processing requests increases by three orders of magnitude as the mean file size for the three classes does also. The other measurements shown in Table 2 which show the same increase are the read calls per connection. This suggests that disk activity explains the increase in elapsed time for processing large requests, as one would expect. One other interesting result in Table 2 is the distribution of network bandwidth among the three request classes. Note that even though the connections per second rate decreases with class number (and requested file size), the bandwidth that each class consumes on the network is about the same (i.e., between 1 and 1.6 Mbps). This is due to the heavy-tailed nature of the file size distribution.

The results from Table 2 suggest that most of the CPU time consumed by the HTTP processes is spent in the kernel. In other words, the task of moving the requested URL from the filesystem to the network is the most expensive part of handling a request. Since both the filesystem and the networking code are in the kernel, one would expect time spent in the kernel to be greater than time in user space. We tested this hypothesis by instrumenting the HTTP processes to call `getrusage` after every 10 requests, and report the user and system time per connection, for the duration of the experiment. These results show that our HTTP processes consume an average of 50 msec of CPU time in the kernel per connection, compared with only 5.2 msec in user space. We'll see later that the kernel module results also demonstrate this high (i.e., 10:1) ratio of system CPU time to user CPU time, for the WWW server as a whole.

5.2 Kernel Module Results

Table 3 shows kernel module (KM) measurements for the workload described above. Recall that KM measures only kernel-level statistics such as overall CPU user and system time, and the rate at which different services are invoked. It therefore does

Table 3 Kernel Module Results for `my_get_kstats` and 30 Clients

<code>cpu_user(%)</code>	9.00	<code>pageins/s</code>	137.42
<code>cpu_sys(%)</code>	90.65	<code>pageout/s</code>	7.94
<code>cpu_idle(%)</code>	0.35	<code>interrupt/s</code>	1011.22
<code>reads/s</code>	5.17	<code>net_interrupt/s</code>	620.43
<code>writes/s</code>	4.90	<code>disk_interrupt/s</code>	289.90
<code>context switch/s</code>	51.66		

Table 4 Kernel Module Results for `my_get_procstats` and 30 Clients

<code>cpu(%)</code>	93.75	<code>started processes</code>	27.55
<code>mem(%)</code>	102.61	<code>running processes</code>	25.01

not separate its measurements according to request class. The most interesting result in Table 3 is that the ratio of system time to user time is high, and is approximately the same as for the HTTP processes monitored by the SM, i.e., 10:1. Within the time spent in the kernel, it is also important to note the relative frequency of certain kernel operations. For example, there are over 100 page-in's, network interrupts, and disk interrupts per second. There are several read calls performed per second. However, there are also a significant number of corresponding write operations per second. These are presumably due to paging activity and logging of HTTP requests.

We have seen a correspondence between SM and KM statistics looking at Table 2 and then Table 3. We also wanted to show a correspondence in the reverse direction. Table 4 shows aggregate process statistics for the HTTP processes, measured in the kernel. Note that the CPU is over 90% utilized, and that memory utilized by the HTTP processes alone is over 100% (which indicates paging activity) . This explains why the CPU user and system times for the HTTP processes (measured by SM) and for the system as a whole (measured by KM) agree. The number of running processes suggests that Apache's process management requires 25 processes to service 30 concurrent connections, given our workload. Finally, the measurements obtained by KM concerning network statistics reported no errors in network interface during the experiments, which is consistent with WebStone results that also reported no errors on the client side.

We also wanted to show that our main conclusions still held when the set of files being requested at the server was sufficiently small to reduce disk activity. A new workload was obtained by dividing the file sizes presented in Table 1 by a factor of 4, but keeping the same number of clients. The results for this workload are shown in Tables 5 and 6. In Table 6, the SM request class sizes are also scaled down by a factor of 4. These results show that, despite reduced disk activity, the ratio of system time to user time remains very high. Both KM and SM results show that over 89% of the CPU time is spent in the kernel. It is also interesting to note the higher connection rate, due to a much faster handling of each request.

Table 5 KM Results for Workload with Smaller Files

cpu_user(%)	9.87
cpu_sys(%)	87.07
cpu_idle(%)	3.06
reads/s	0.79

Table 6 SM Results for Workload with Smaller Files

user time(ms)/conn	23.97
system time(ms)/conn	2.83
(aggregate) conn/s	25.74
(aggregate) Mbits/s	3.45

The validation of the results collected by KM and SM was done through comparison with similar measurements obtained through the /proc filesystem and WebStone, respectively. The differences between them are less than 1% [Almeida Almeida *et al.*, 1996].

5.3 Effect of Keeping TCP Connections Open

We wanted to use Webmonitor to measure the effect of keeping TCP connections open on our Web server. Recall that this is a requirement of TCP, to guard against old data being received by a new connection. To isolate this effect, we reproduced the experiments described above, but changed Linux's TCP implementation to close connections without spending any time in the TIME_WAIT state. Although such a TCP implementation is not "legal," this modification allowed us to show the effect of keeping connections open on server behavior. In a legal implementation, the TIME_WAIT state is entered to catch and discard packets from a closed connection, that were retransmitted by a client. The usual holding time in this state is 60 seconds, after which the connection is closed (put in the TCP_CLOSE state). It has been observed by others [Mogul, 1995a, Mogul, 1995c, Padmanabhan *et al.*, 1994] that the holding time in the TIME_WAIT state is a possible performance problem for WWW servers, however, we are the first to quantify this and give some insight into possible causes.

Table 7 Number of Connections in TCP States (KM)

TCP State	TIME_WAIT = 0	TIME_WAIT = 60 sec
ESTABLISHED	29.14	29.74
TIME_WAIT	0	941.84
CLOSE	64.45	35.03

Table 7 gives the average number of connections seen in different TCP states. Although TCP actually has 11 states, the number of connections in the other 8 states was zero or negligible. The most interesting number in Table 7 is the large number (over 900) connections in the TIME_WAIT state, when its holding time is 60 seconds. These results are consistent with those in [Mogul, 1995a, Mogul, 1995c]. Fortunately, because of the large number of TCP connections that may be open at the same time, Linux uses a hashed lookup table with a single entry cache to store its connection descriptors. Such an implementation guards against connection descriptor lookup times that increase linearly with the number of open connections (a common

mistake in older TCP implementations). It is also interesting to note that more time is spent in the closed state (TCP_CLOSE), than in the state where the connections are actually performing useful work (the ESTABLISHED state).

Table 8 Server Module Results for TIME_WAIT of 60 with 30 Clients

TIME_WAIT = 60 sec	Class 1	Class 2	Class 3
conn/s	16.40	0.88	0.07
Mbits/s	1.55	1.17	1.19
reads/conn	0.03	2.67	34.67
CPU parsetime(ms)/conn	4.78	4.81	3.81
CPU processingtime(ms)/conn	18.75	150.56	2231.35
CPU logtime(ms)/conn	5.28	6.99	9.10
elapsed parsetime(ms)/conn	23.45	21.80	6.20
elapsed processingtime(ms)/conn	155.81	3789.95	60578.90
elapsed logtime(ms)/conn	774.31	940.35	827.42

Table 9 Server Module Results for TIME_WAIT of 0 with 30 Clients

TIME_WAIT = 0	Class 1	Class 2	Class 3
conn/s	24.64	1.25	0.10
Mbit/s	2.33	1.66	1.87
reads/conn	0.02	2.24	33.91
CPU parsetime(ms)/conn	2.41	2.34	2.41
CPU processingtime(ms)/conn	15.84	101.52	1373.71
CPU logtime(ms)/conn	6.09	7.07	7.05
elapsed parsetime(ms)/conn	8.62	6.26	5.83
elapsed processingtime(ms)/conn	77.89	2519.07	37417.10
elapsed logtime(ms)/conn	559.37	559.42	530.52

To understand the impact this large number of TIME_WAIT connections has on server performance, we first looked at results from the SM. Tables 8 and 9 show SM results for 30 clients using a TIME_WAIT time of 60 seconds and 0, respectively. The results for latency and throughput (conn/s and Mbit/s) show a dramatic difference in performance. Having a TIME_WAIT time of 60 seconds makes all the work that a server performs (i.e., parsing, processing and logging an HTTP request) take longer. This is true in terms of both CPU time and elapsed time. For example, the CPU time to process a large (i.e., Class 3) request is two times greater for a TIME_WAIT time of 60 seconds than for a TIME_WAIT time of 0.

As a consequence of the increase in latency when the TIME_WAIT time is set to 60, throughput decreases significantly. The performance penalty for the server can be seen by a 33% higher number of connections serviced (and Mbits sent) per sec-

ond, when the `TIME_WAIT` time is 0. It is also interesting to examine the resources consumed at the server. Our KM results for both configurations (not presented here) show that the consumption of all resources is roughly the same. Both CPU and memory utilizations are over 90%, indicating that the server is saturated in both experiments. Furthermore, roughly the same number of HTTP processes are used to handle the higher request rate when no time is spent in the `TIME_WAIT` state.

These results clearly show that the impact of the `TIME_WAIT` holding time is twofold. First, although roughly the same number of HTTP processes are active at the same time, these processes are able to handle a lower number of requests. Second, that these processes consume more CPU time to serve each request. This is, of course, only part of the answer to the larger question of whether memory, I/O, or the CPU is the bottleneck for WWW servers.

6 CONCLUSION

Server performance has become a crucial issue for improving the overall performance of the World-Wide Web. This paper describes Webmonitor, a tool for evaluating and understanding server performance, and presents new results for a realistic workload. These results emphasize the important role of operating system and network protocol implementation in determining Web server performance.

Webmonitor measures activity and resource consumption, both within the kernel and in HTTP processes running in user space. Webmonitor is implemented using an efficient combination of sampling and event-driven techniques that exhibit low overhead (less than 4%). We demonstrate the utility of Webmonitor by measuring and understanding the performance of a Pentium-based PC acting as a dedicated WWW server. Our workload, generated by WebStone, uses a file size distribution with a heavy tail. This captures the fact that Web servers must concurrently handle some requests for huge files and a large number of requests for small files.

Our results show that in a Web server saturated by client requests, up to 90% of the time spent handling HTTP requests is spent in the kernel. Furthermore, keeping connections open, as required by TCP, causes a factor of 2 increase in the elapsed time required to service an HTTP request. This increase in latency (caused by a high number of connections in the `TIME_WAIT` state) is accompanied by a 33% reduction in server throughput.

Although this paper provides an important understanding of World-Wide Web server behavior under heavy load, the picture is far from complete. There is still the question of whether memory, I/O, or the CPU is the bottleneck for Web servers. The answer to this question will probably depend on the nature of the workload, however, there will continue to be a demand for server architectures that perform well for heterogeneous workloads. This suggests the need for new operating system and network protocol implementations that are designed to perform well when running on Web servers.

ACKNOWLEDGEMENTS

We would like to extend special thanks to Pei Cao and Gideon Glass at the University of Wisconsin for insightful discussions about this work. We would also like to

thank Jeff Mogul at DEC WRL, Erich Nahum at IBM Watson, Wagner Meira at the University of Rochester, and the anonymous reviewers for improving the content and presentation of this paper.

REFERENCES

- [1] Almeida, J. M., Almeida, V. and Yates, D.J. (1996), Measuring the behavior of a World-Wide Web Server, Technical Report 96-025, Boston University.
- [2] Almeida, V., Bestavros, A., Crovella, M. and Oliveira A. (1996), Characterizing reference locality in the WWW, *Proceedings of IEEE-ACM PDIS'96*.
- [3] Arlitt, M. and Williamson, C. (1996), Web server workload characterization, *Proc. of the 1996 SIGMETRICS Conference on Measurement and Modeling of Computer Systems*.
- [4] Birman, K. and Renesse, R. (1996), Software for reliable networks, *Scientific American*.
- [5] Cockcroft, A. (1996), Watching your web server, *SunWorld Online*, URL: <http://www.sun.com/sunworldonline/swol-03-1996/>.
- [6] Crovella, M. and Bestavros, A. (1996), Self-similarity in world wide web traffic: Evidence and possible causes, *Proc. of the 1996 SIGMETRICS Conference on Measurement and Modeling of Computer Systems*.
- [7] Lai, K. and Baker, M. (1996), A performance comparison of UNIX operating systems on the Pentium. *Proceedings of the 1996 USENIX Conference*, San Diego, CA. USENIX.
- [8] McGrath, R., Kwan, T. and Reed, D. (1995), NCSA's world wide web server: Design and performance, *IEEE Computer*.
- [9] Menasce, D., Almeida, V., Dowdy, L. (1994), *Capacity Planning and Performance Modeling*, Prentice Hall, Englewood Cliffs.
- [10] Mogul, J. C. (1995), Network behavior of a busy Web server and its clients, Research Report 95/5, DEC Western Research Laboratory.
- [11] Mogul, J. C. (1995), Operating system support for busy Internet servers, *Proceedings of the Fifth Workshop on Hot Topics in Operating Systems*.
- [12] Mogul, J. C. (1995), The case for persistent-connection HTTP, In *SIGCOMM Symposium on Communications Architectures and Protocols*, **299-313**, Cambridge, MA. ACM
- [13] Padmanabhan, V. N. and Mogul, J. C. (1994), Improving HTTP latency, In *Proceedings of Second WWW Conference '94: Mosaic and the Web*, **995-1005**, Chicago, IL.
- [14] Robinson, D. and the Apache Group (1995), *APACHE – An HTTP Server, Reference Manual*, URL: <http://www.apache.org>.
- [15] Somin, Y., Agrawal, S. and Forsyth, M. (1996), Measurement and analysis of process & workload CPU times in UNIX environments, *Proceedings of the CMG'96*.
- [16] Trent, G. and Sake, M. (1995), *WebStone: The First Generation in HTTP Server Benchmarking*, URL: <http://www.sgi.com/Products/WebFORCE/WebStone/paper.html>
- [17] Welsh, M. (1994), *The Linux Bible*, Yggdrasil Computing Incorporated, 2nd edition.

A Framework for ATM Testing

J.B. Micheel, C. Tittel, J. Tiemann

ATM Test Center and BATES Projects

GMD FOKUS

Hardenbergplatz 2, 10623 Berlin, Germany

Phone: +49-30-25499-268, Fax: +49-30-25499-202

Email: {micheel,tittel,tiemann}@fokus.gmd.de

Abstract

ATM technology is of very complex nature. Different views exist on how to get implementations of this network technology robust and interoperable. This paper describes a unified approach to ATM testing (i.e. interoperability, conformance, performance, debugging, monitoring, etc.), which reflects in a modular set of building blocks that can be combined for a particular test.

Keywords

ATM testing, monitoring, troubleshooting

1 HISTORY AND MOTIVATION

GMD FOKUS' first ATM testbed dates back to June 1992 when 2 SPARCstation 2 where connected with SBA-100's from FORE systems. Some tests where made to evaluate the state of ATM development in practice. The first switch, a FORE ASX-100, arrived in December 1992 and was one of the first workgroup switches being sold in Europe.

Since then the network has grown permanently (Micheel, 1997) and now consists of more than a dozen switches from various vendors, a couple of interworking units and around 50 workstations directly connected to ATM.

The FOKUS inhouse ATM network immediately joined BALI (Berlin ATM LAN Initiative) , one of the first metropolitan area ATM networks. It is based on dark fibers of the former BERKOM high-speed testbed with major sites at DeTeBerkom (Deutsche Telekom Group), GMD FOKUS and Technical University of Berlin.

A network of such size (with DeTeBerkom there are more than 100 networked entities) demands for equipment to test and monitor individual devices and the network in its entirety, the more, since a number of extremely challenging demonstrations are run across this net. As an example, BALI now has transcontinental links into north america (MAY testbed, 155 MBit/sec trunk speed) and south-east asia (Singapore, 2 MBit/sec). GMD FOKUS is equipped with an HP 75000 BSTS, a Tekelec ChameleonOpen and a number of smaller testers. An ATM tester project was set out - resulting in the BATES (Berkom ATM Technology Evaluation System) platform. The ATC (ATM Test Center) project at DeTeBerkom and FOKUS evaluated a large number of ATM testers currently being available commercially (Kühn, 1996). The result of these efforts remains unsatisfying:

- ATM testers are usually grouped into protocol testers / decoders and “high-end” testers dedicated to performance measurements at the ATM layer. There is no single tester providing both in a satisfactory way.
- ATM testers come with a fixed set of software. Test software for a certain ATM feature usually becomes available with a delay of one year after first implementations showing up.
- ATM testers provide a poor environment to describe test scenarios. Some provide a C programming library interface, which is clumsy and inherently tester-dependend. There also is a very limited number of users who would go into programming a tester just to customize it for a certain test case.
- ATM testers are generally expensive devices. While organizations like FOKUS are able to spend money for one or two dedicated systems, these testers are far from being the workhorse for any kind of testing for a large number of researchers doing their job in parallel.
- ATM testers do not allow for testing or monitoring at different layers (e.g. ATM, AAL, TCP/IP, ...) in parallel. So when debugging a TCP session by looking at the data packets you’ll not be able to see cell lossage due to traffic contract violation causing troughput degradation, which you are, in fact, hunting for.
- High-end testers are usually non-portable, at least to carry them with you when being at a demonstration or outhouse testbed, where a tester is of much help “in the case something breaks”.

In the fire of these constraints the idea of workstation-based tools was born. A simple ATM adaptor provides for access to the net or device under test. All else is done in software. An example is ATMCell - a tool based on FORE's SBA-100 hardware, which allows for cell-level protocol en- and decoding and microsecond-granularity cell delay measurements. With this tool switching delays of local ATM switches could be measured for less than US\$1000. Higher layer decoding and debugging in multi-vendor environments highlighted bugs in protocol implementations (EC LACE project). However, the approach was rejected for both technical and esthetic reasons. Implementation, user interface and test model would not run transparent on other hardware providing similiar services (i.e. HP BSTS). Then, the power of the user interface would not scale with our demands for configurability. The idea for CellFile - a stream interface to access ATM-level test hardware and to combine ATM-level tools - was born.

With ATM devices gaining a certain degree of useability, more and more functionality became available at AAL and higher layers. Standard ATM adaptor cards implementing AAL3/4 and 5 in firm- or hardware provided a basis to design a number of important tools: sigdump (Tittel, 1995) (Tittel, Micheel, 1996), lanedump (Mochalski, Barth, 1996) (Mochalski 1996) and mpoadump. Again, to keep tools free from handling ATM adaptor and API specifics, separation of functionality became a necessity. PacketFile - a stream interface for variable-length messages - would serve our needs.

Running an ATM network at this time is not just challanging due to implementations still getting adult, but because there are no tools to identify the source of trouble, thus forcing a network administrator to "fix" problems heuristically by guessing and rebooting switches and workstations. A troubleshooting system would be very beneficial. FOKUS has started to instrument it's own network in order to overcome such problems and within the ATM test framework to develop the necessary set of tools for private network troubleshooting. A central role play decoding tools (sigdump, lanedump, etc.), but these tools require as input the link state (channel activity plus higher layer protocol information). We consider to develop a number of state agents holding this information for later use by decoders. These agents provide their information via SNMP. An umbrella agent (called CIA) gives a unified view onto these standalone agents. sigagent (Sträuber, 1996) is a first implementation that holds link state based on information of the UNI signalling flow. (Micheel, Tittel, 1996) provides more information about the approach.

The current framework thus stems its power from a few paradigms:

- Hardware is used to convert a CPU-processable representation of test data (read: CellFile, PacketFile) from and to the wire.

- Any semantics describing a certain test scenario are assembled out of atomic modules providing basic features. Examples are filtering or statistics modules. These modules are currently implemented as CPU-instances (processes or streams modules), but the abstraction allows for later implementation in firm- or hardware. In any case (for now), the CPU should be kept in the data stream to provide a maximum of flexibility when designing and assembling a test scenario.
- Writing new modules shall be lots of fun. The programming environment must be simple and easy to understand, forcing human cycles to be spend on solving the problem instead of fighting against the framework.
- Keeping the CPU in the data stream implies destroying unnecessary bandwidth as early as possible and generating useful information instead. An example would be an ATM layer statistics module taking any cell on the wire and producing once a second a summary about activities.
- At the very end of the chain of modules there is always a human sitting in front of the tester who is most often interested to get a very simple answer. The framework shall give this answer by allowing to formulate questions.

2 CONCEPTS

The core idea of the framework is to identify a minimal set of building blocks that satisfy complex testing. We define ATM testing as the superset of all kinds of activities ensuring proper operation of an ATM-based network or system.

Assembling a test case out of modular building blocks requires defining an interface between those modules. Keeping the number of different interfaces low ensures a maximum of flexibility when combining tools. We have, at current, decided to stick to 3 simple protocols: **CellFile**¹, **PacketFile**² and **SNMP**.

The CellFile (CF) interface is based on the definition of an ATM cell (ATM Forum UNI) (ATM Forum B-ICI), extended with a timestamp and layed out (padded) in a way that allows for easy processing by 32bit CPU's (see also Cooper et.al., 1991). The Cell File is the core protocol of the framework, since anything in an ATM network happens as ATM cells being transmitted over bidirectional links. Higher layer information may be recovered from or encapsulated in these cells.

¹. The term "File" is a misnomer, it really is a stream interface format.
The name remains for historical reasons.

². see 1.)

Timestamping captured cells is a way of data compression (elimination of idle or unassigned cells) and a representation of the traffic profile - essential feature for a network technology with complex descriptions of application behaviour and Quality-of-Service requirements.

A special **CellHeader** (CH) format allows for analysis and testing at higher data rates by stripping payload information, thus reducing the bandwidth resp. storage requirements to 20% compared to the standard CellFile. This format is very convenient for ATM layer performance measurements, where payload contents is irrelevant. It allows to build inexpensive hardware for capturing and transmission of traffic profiles at today's standard link rates (155 MBit/sec) (Micheel et. al., 1996). (TANYA's I-want-it-all design addresses the problems with other ATM testers described in the first section of this paper.) The header fields and the toolset for CH streams are identical to CF, so everything said about CF includes CH in the rest of this paper.

The second major interface format is PacketFile (PF), a sequence of variable-length messages marked with a timestamp. We found the format, in which data packets are stored by tcpdump (McCanne, Jacobsen, 1993), to nicely fit our requirements. Tcpdump's traces may contain header-only information (a predefined number of bytes starting at byte #0), again reducing the number of bytes to be processed by PF tools and keeping the storage size of captured profiles low.

Both cell and packet file formats hold a timestamp per unit (cell or packet) for different purposes. Very important is inter-unit spacing to retain the traffic profile of the original stream in the captured representation or to control the behaviour of a stream to be sent out. For this purpose relative timestamps would be best. A second goal is to allow for a later merge or comparison with other streams: both directions of a communications link might be monitored to look at request-response performance or a 2-point measurement could allow for cell transfer delay and delay variation analysis. For this kind of tests absolute timestamps are a better choice. Instead of deciding for a single method we currently support both and use whatever coding is best for the scenario in question. The downside of this is that modules have to support both formats, which makes implementation and usage harder. The role and usage of timestamps is context-dependent. There are also performance and flexibility tradeoffs when choosing a certain format: the higher the accuracy the more bits you need to describe larger intervals (thin traffic). More overhead bits for timestamping are a performance degradation for the toolset. 32 bits are very easily processed by today's processors, more bits result in a significant reduction. At current, the final choice of the timestamp format is still a research topic.

A SNMP (Case et. al., 1991) MIB is the database of choice to hold information gathered by monitoring an instrumented network. First, SNMP is a fairly simple and

robust protocol. Second, because of ASN.1 BER encoding, it is able to transport almost any kind of data. Agents to be written capture and display data of various kinds: ATM level activity information, measured traffic flow parameters, ATM signalling and routing state, human input and other. Third, data can be conveniently substructured in MIB's. The umbrella SNMP agent (CIA) then allows to combine different agents to give a link-complete view, or to assemble a network view by collecting the information from different monitoring points of the same agent (e.g. signalling). CIA could also implement more sophisticated functionality, such as filtering or scoping, which are not present as SNMP protocol features.

3 EXAMPLES

In this section we'd like to illustrate the broad range of configurability we intend to achieve with modules within our framework. any of the pictures TANYA might be replaced by a similiar hardware for ATM-level access, the combination of TANYA with AAL5 segmentation and reassembly with a standard of-the-shelf workstation ATM adaptor. In fact, since TANYA is non-existing hardware (at the time of this writing), the entire toolset as currently exploited is based on a number of different platforms. Other access points for higher-level analysis (AAL, IP, LANE, ...) might be standard debugging hooks in network driver implementations, such as Sun's NIT, BPF or DLPI.

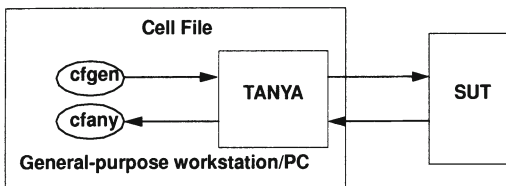


Figure 1 Performance measurement scenario

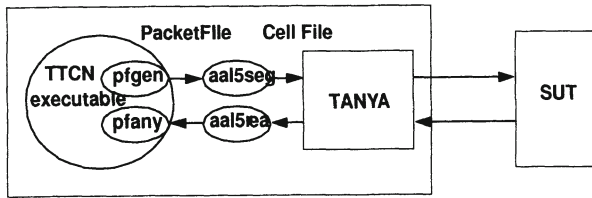


Figure 2 Conformance test scenario

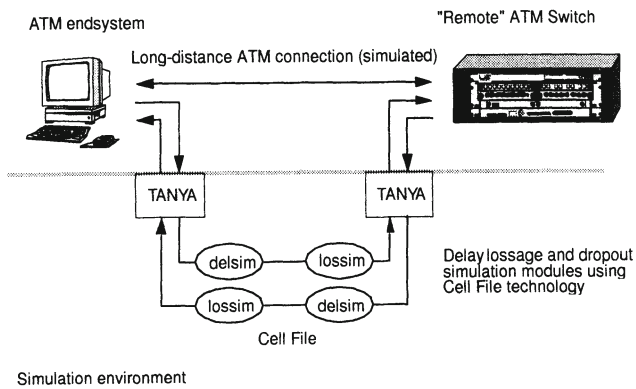


Figure 3 On-the-fly simulation of network behaviour

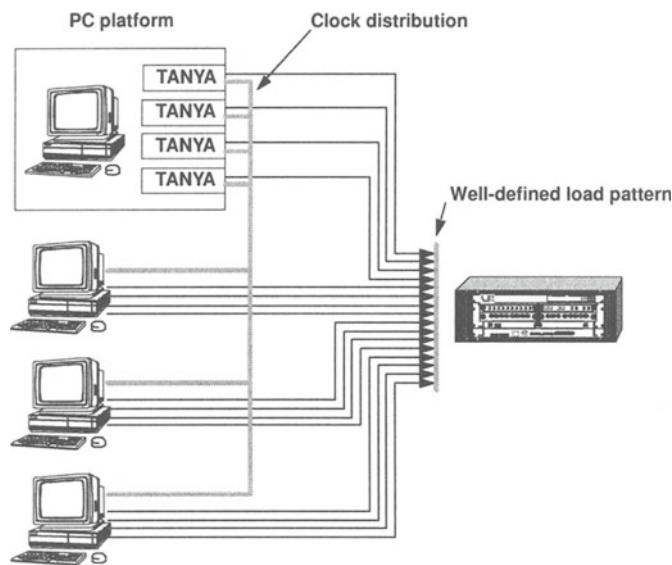


Figure 4 Large-scale distributed testing

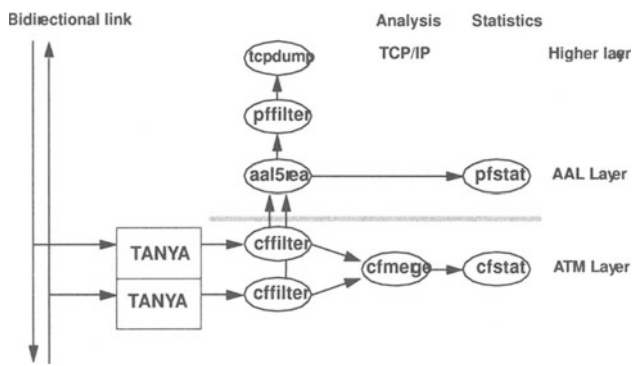


Figure 5 Network monitoring and troubleshooting

4 DEVELOPMENT STATE

At current, a number of protocol decoding tools have been implemented (UNI signalling, ILMI, LAN Emulation). The packet file toolset fitting with these decoders implements ASCII conversion, filtering based on the Berkeley Packet Filter, merging of PF streams. CF tools have been written for a number of experimental purposes (generators, analyzers, etc.), but are still missing the strong split of functionality and cleancut. A first agent is available - sigagent. The instrumentation of the FOKUS and BALI local and MAN ATM networks is in progress, a photonic switch has been obtained to join measurement points independent of the various physical layer speeds to be run on multimode and singlemode fiber. A heterogenous multi-vendor testbed is envisioned for the near future.

The next steps to be taken are:

- to enrich the toolset trough decoders for P-NNI, MPOA, B-ISUP (in this order)
- to develop cheap and powerful access hardware - TANYA
- to evaluate scalability for the CPU-based approach of ATM testing
- to develop a clean set of CF tools
- to evaluate the feasibility of the framework for permanent ATM monitoring, accounting and billing.

There are plans to publicly release parts of the software (binaries and/or sources) as **FAST** - FOKUS ATM ToolSet - but this hasn't been our primary goal in the past, so the release is delayed.

5 ACKNOWLEDGEMENTS

We owe special thanks to our colleague Falk M. Schulz, now with DeTeBerkom, for co-designing an early draft of the framework and implementing first versions of the CF tools as part of his ATMCell test tool. Thanks to all other colleagues of the ATC and BATES projects for enriching the number of miscellaneous CF and PF tools.

6 REFERENCES

- ATM Forum, Technical Committee, Broadband Inter Carrier Interface (B-ICI) Specification Version 1.0
- ATM Forum, Technical Committee, User-Network Interface (UNI) Specification Version 3.1, September 1994.

- Case J.D., Fedor, M.S., Schoffstall, M.L. and Davin J.R. (1990) A Simple Network Management Protocol, *Internet Engineering Task Force (IETF) Request For Comments (RFC) 1157*, May 1990.
- Cooper, E., Menzilcioglu, O., Sansom, R. and Bitz F. (1991) Host Interface Design for ATM LAN's, in *Proc. of the 16th Conference on Local Computer Networks*, IEEE Computer Society, p. 247-258.
- Kühn, K. (1996) Testwerkzeug für ATM Signalisierung, Diplomarbeit an der Technischen Universität Dresden, DeTeBerkom und GMD FOKUS.
- McCanne, S. and Jacobsen, V. (1993) The BSD Packet Filter: A New Architecture for User-Level Packet Capture, in *Proc. of the 1993 Winter USENIX Conference*, January 25-29, 1993, San Diego, CA.
- Micheel, J., Schmidt, R. and Tiemann J. (1996) TANYA - Yet ANother ATM Tester or I-want-it-all, GMD FOKUS research paper, August 5th, 1996.
- Micheel, J. and Tittel, C. (1996) ATMSnoop - Watching The Cells Go By, in *Proc. of the 8th IEEE Workshop on Local and Metropolitan Area Networks*, Potsdam, Germany, 25.-28. August 1996.
- Micheel, J. (1997) The FOKUS ATM and BALI Networks. Informational material, GMD FOKUS.
- Mochalski, K. and Barth, M. (1996) LAN Emulation. ATC Technical Report #2, GMD FOKUS.
- Mochalski, K. (1996) Entwicklung und Implementierung eines LAN Emulation Testtools zur Evaluierung von ATM Komponenten, Diplomarbeit an der Universität Leipzig und GMD FOKUS.
- Sträuber, P. (1996) Entwicklung und Implementierung eines Signalisierungs-SNMP-Agenten, Diplomarbeit an der Universität Leipzig und GMD FOKUS.
- Tittel, C. (1995) Entwicklung und Implementierung eines Signalisierungs-Testtools zur Evaluierung von ATM Komponenten, Diplomarbeit an der Universität Leipzig und GMD FOKUS.
- Tittel, C., and Micheel, J. (1996) UNI Signalling. ATC Technical Report #1, GMD FOKUS.

7 BIOGRAPHY

Jörg Micheel received his Dipl.-Ing. at the Novosibirsk Institute for Electrical Engineering, Russia. In 1992 he started working at GMD FOKUS as a systems programmer for the BERMATE high-speed ATM adaptor. Since 1995 he leads the ATM Test Center joint project of DeTeBerkom and GMD FOKUS. He has been administering the FOKUS ATM and BALI networks since the early beginnings.

Currently, his research focuses on smart solutions for creating a powerful and robust networking service.

Dipl.-Inform. Carsten Tittel joined FOKUS as a student at Leipzig University. Since 1996 he's a full-time researcher involved in the European Community ACTS INSIGNIA project, where he works on functional testing of protocol implementations.

Dipl.-Ing. Jens Tiemann brood over BERMATE as a hard-worker. Since 1995 he led the BATES project and pursued the integration of CellFile technology. In his spare time he spend cycles on making things more efficient and co-designed TANYA, which is now his primary project.

PART THREE

Multimedia Traffic

The Effect of Various ATM Switch Architectures on VBR Video Performance

Rose P. Tsang - rtsang@ca.sandia.gov Jenwei Hsieh and David H.C. Du
Sandia National Laboratories University of Minnesota
Livermore, CA 94551 Minneapolis, MN 55455

Abstract

One of the most important components of an Asynchronous Transfer Mode (ATM) network is the switch. Switch design is not a part of the ATM standards so vendors use a wide variety of techniques to build their switches. In this paper, we present experimental results of switching and multiplexing real-time Variable Bit Rate (VBR) digital video traffic (JPEG, MPEG-1, and MPEG-2) through two different ATM switch architectures. Real-time VBR traffic, such as digital video, is particularly interesting due to its high demands in terms of bandwidth, real-time delivery and processing requirements. Our experiments show that the 'fastest' switches, i.e., lowest latencies, do not necessarily perform better when transmitting VBR video. The impact of the high speed network components' characteristics, such as switch fabric architecture, buffering strategies, and higher layer transport protocols (i.e., UDP, TCP/IP), are illustrated through the experimental results.

Keywords

Asynchronous Transfer Mode (ATM), digital coded video, JPEG, MPEG, switch architectures, buffering strategies.

1 INTRODUCTION

One of the most important components of an Asynchronous Transfer Mode (ATM) network is the switch. An ATM switch must accept asynchronous and synchronous traffic, as well as connection-oriented and connection-less traffic. The speed at which a switch relays cells within a pre-specified cell loss probability bound is considered crucial; queueing and switching delay must be minimized. The switch processor must also be capable of handling high speed ports and aggregate throughputs of at least a couple gigabits per second.

Do high switch throughputs and low switch latencies translate into superior performance for all traffic types ? The ATM standard is expected to serve as the transport mode for a wide spectrum of traffic types with varying performance requirements. In this study, we are interested in how well ATM switches can support real-time Variable Bit Rate (VBR)

traffic. We present experimental performance results of real-time VBR traffic, namely digital compressed video, over two different ATM switch architectures. The experimental results are followed by a discussion of the results in relation to the capabilities and features of the switch architectures and network environment. We are interested in digital video because it is expected to be one of the predominant type of multimedia traffic transmitted on future high speed networks [Cole (1993), Pancha (1994)]. Video requires the continuous periodic delivery of data in order for the user not to perceive a degradation in the service quality (e.g., jerkiness due to the erratic delivery of video frames). Some data loss may be tolerated but the data must be delivered within a bounded amount of time. This is considered to be one of the most difficult data types to transport due to its real-time requirements as well as its dynamic variability of bandwidth requirements [Cole (1993), Tsang (1996)].

The type of video quality expected by today's users require very large network transmission speeds. Uncompressed speeds of broadcast-quality NTSC video and studio quality NTSC video are about 120 Mbits/sec and 216 Mbits/sec, respectively [Cole (1993), Lyles (1992)]. Fortunately, recent advances in compression techniques [Cole (1993), LeGall (1991), Wallace (1991)] make the transmission of these high quality video media feasible. These compression techniques take advantage of the characteristics (or limitations) of the human visual system, to achieve lossy, yet visually lossless, compressed images and video. The two main video compression standards are commonly known as Joint Photographic Experts Group (JPEG) [Cole (1993), Wallace (1991)] and Moving Picture Experts Group (MPEG) [Cole (1993), LeGall (1991), Pancha (1994)]. JPEG is a still-image compression standard which can produce visually lossless compression with ratios up to 10 to 1. The MPEG motion video standard was designed to support full motion video at compression ratios up to 200 to 1. Bandwidth requirements for MPEG-2 range from 4 to 15 megabit per second, for a stream of $640 \times 480 \times 24$ pixels per frame (24 bit color), 30 frames per second digital NTSC-quality video [Cole (1993)].

The goals of our study were the following:

- We sought to measure the ability of an actual ATM local area network to support real-time variable bit sources, i.e., bursty periodic data. Real-time variable bit rate traffic tolerates a small degree of loss but strict timing requirements on frame delivery must be met. Moreover, frame sizes may vary greatly even between consecutive frames. Previous work on the performance of packet switch networks to support variable bit rate sources have used either analytical means or simulation models to predict performance [Heffes (1986), Norros (1991)]. However, accurate analytical models which capture the time-varying correlated nature of the stochastic process that model video streams are usually intractable [Maglaris (1988)]. It is also obvious that even in simulation models it is not possible to capture all aspects (or even most) of an actual distributed system. In a distributed system, there exist many components, hardware and software, whose complex interactions cannot be naturally captured or predicted by a fixed model.
- It is likely that video transmission will be one of the dominant media types involved in distributed multimedia systems. Most non-trivial video-based multimedia applications entail the network support of many multiplexed coded video streams. Given an ATM environment, with a known maximum achievable throughput and delay, how many

typical coded video streams (e.g., MPEG-1, MPEG-2, JPEG) can be supported within reasonable loss and jitter bounds ?

- We sought to determine the effect of switch architecture on the performance of digital video transmission. The Fore Systems ASX-200 and Cisco A100 Hyperswitch switch architectures provided an interesting basis for comparison of ATM switches architectures. Both switches rely upon Time Division Multiplexed (TDM) shared buses as the connecting switch fabric between input and output ports. However, both implement very different buffering strategies. We will present the effect of these different buffering strategies in our experiments.
- The Transmission Control Protocol (TCP/IP) and User Datagram Protocol (UDP/IP) suite of protocols are widely used today, and are very likely, during the initial deployment of ATM networks, to remain the dominant suite of protocols used in local and wide area network computing. How well is each suited for transmitting multiplexed video streams ?

2 TESTBED ENVIRONMENT

The testbed environment consists of the following components:

Eight IBM RISC System/6000 (RS/6000) workstations. Each RS/6000 has a 66 MHz POWER2 processor. The operating system used was AIX 3.2.5. Each RS/6000 was equipped with an IBM Turboways 100 ATM Adapter (TURBOWAYS adapter) [IBM (1994)]. The TURBOWAYS adapter uses an onboard 25 MHz Intel i960 processor. The adapter provides dedicated 100 Mbps full-duplex connectivity using PVCs, and Direct Memory Access (DMA) capabilities.

A Fore Systems Forerunner ASX-200 ATM switch. The ASX-200 local ATM switch is Fore System's successor to the ASX-100 switch. Like the ASX-100, it is based on a 2.4 Gigabits per second switch fabric (TDM bus) and a RISC control processor. The Fore ASX-200 switch architecture provides output buffering but no input buffering (see Figure 1).

Its features include output buffers with a 13,312 ATM cell capacity (per 2 output ports), dual leaky bucket policing, 'smart' buffers (per-VC queueing, multiple service priorities, dynamic buffer allocation), packet level discard and congestion control through the Explicit Forward Congestion Indicator.

A Cisco A100 Hyperswitch. The Hyperswitch is also based upon a 2.4 Gigabit per second TDM bus-based switch fabric. Its cells are routed through a combination of input and output buffers (see Figure 1). The input buffer size (per port) is 2450 ATM cells and the output buffer size (per port) is 50 ATM cells. Cells traverse through the switch fabric via a credit scheme which controls the admission of cells from the input buffers to the output buffers. In credit-based schemes, cells are never admitted to a network component unless that component is forwarded a 'credit' [Fan (1994)]. In the Hyperswitch, cell loss never occurs at the output buffers because before a cell may be admitted to the output buffer, the output buffer must send a signal (or credit) to the input buffer to let it know that it has available buffer slot(s). The Hyperswitch's switch architecture was based upon the "Expandable ATOM switch (XATOM)" design proposed at NEC [Fan (1994)].

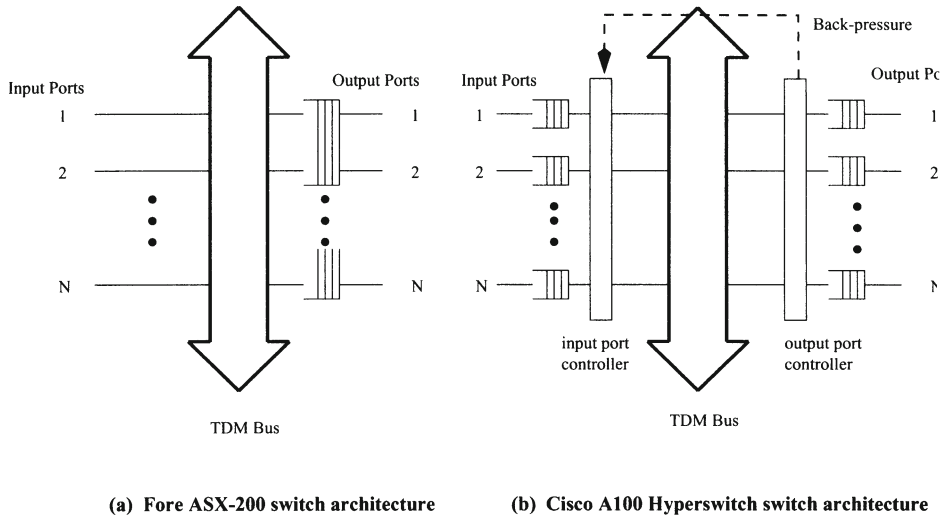


Figure 1 Switch Architectures

100 Mbts/sec TAXI links interconnecting the RS/6000 hosts to the ATM switch. The topology of our experimental testbed consisted of an ATM switch (either the Fore or the Cisco) with direct TAXI connections to the 8 RS/6000 hosts. Connectivity between hosts was via one hop through the ATM switch.

The video data consisted of the following:

MPEG. The input video stream for the MPEG codec was a 3 minute 40 second digitized from laser disc with a frame resolution (similar to NTSC broadcast quality) of 512×480 pixels. This particular Star Wars sequence was chosen because it contained a mix of high and low action scenes. The interframe to intraframe ratio was 16. The quantizer scale was 8. For these parameters, the image quality was judged to be good (constant) through the entire sequence of frames. The coded video was captured at 24 frames/second. The mean bit rate for the MPEG-1 sequence was approximately 1.5 Mbts/sec; the bit rate varied from about 0.3 Mbts/sec to 4.5 Mbts/sec. The mean bit rate for the MPEG-2 sequence was approximately 5.0 Mbts/sec; the bit rate varied from about 0.8 Mbts/sec to 10.5 Mbts/sec.

JPEG. The input video stream for the JPEG codec was also a sequence from the Star Wars movie. The movie was input from a VCR and processed by a video card from Parallax Graphics, Inc., inside the Sparc 2. The frame resolution was 512×480 pixels. The quantization factor was 400. The image quality was judged to range from fair to marginal. The coded video was captured at 15 frames/second. The mean bit rate was approximately 2.3 Mbts/sec; the bit rate varied from about 2.0 Mbts/sec to 2.5 Mbts/sec.

It is important to note that the input sequence we chose contains high as well as low motion scenes. Hence the performance results cannot be compared to all types of video

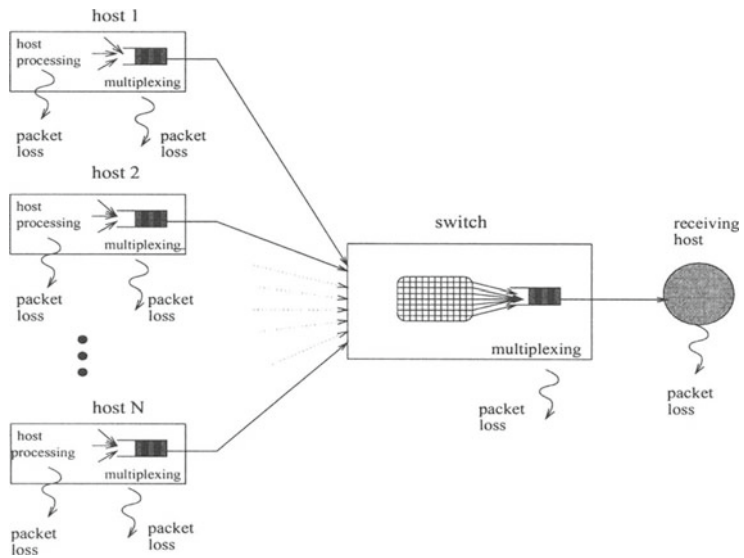


Figure 2 N senders to 1 receiver configuration

sequences. For instance, a video sequence generated from a video-conferencing application would contain, on average, smaller frames, since in most video-conferencing sequences there is little motion and few, if any, scene changes. Thus, better performance would be expected if such a sequence were used. We chose a video sequence with a high-level of motion in order to stress test the ATM network's ability to support bursty data.

We used the following configurations to measure how well the Fore ASX-200 switch and the Cisco Hyperswitch handled contention at the switch fabric, i.e., cross-traffic (**N senders to N receivers configuration**), and contention at the output ports (**N senders to 1 receiver**).

- **N senders to 1 receiver.** In this configuration, N RS/6000s simultaneously injected multiple video streams through the switch to a single receiver. See Figure 2.
- **N senders to N receivers.** In this configuration, N RS/6000s simultaneously injected multiple video streams through the switch to another disjoint set of N RS/6000s. See Figure 3.

The following terminology and performance metrics are used throughout the remainder of the paper.

The **average send time** of a frame is the time the first packet of a frame is transmitted to the time the first packet from the next consecutive frame is transmitted. When transmitting video frames at 24 frames per second, the average send time is $1/24 = 41.67$ milliseconds. The **received interval** is the time between the receipt of an entire frame (the last packet in a frame) and the receipt of the next entire frame. This metric is used to compute **jitter** - the interarrival delay between consecutive frames. The **fixed frame**

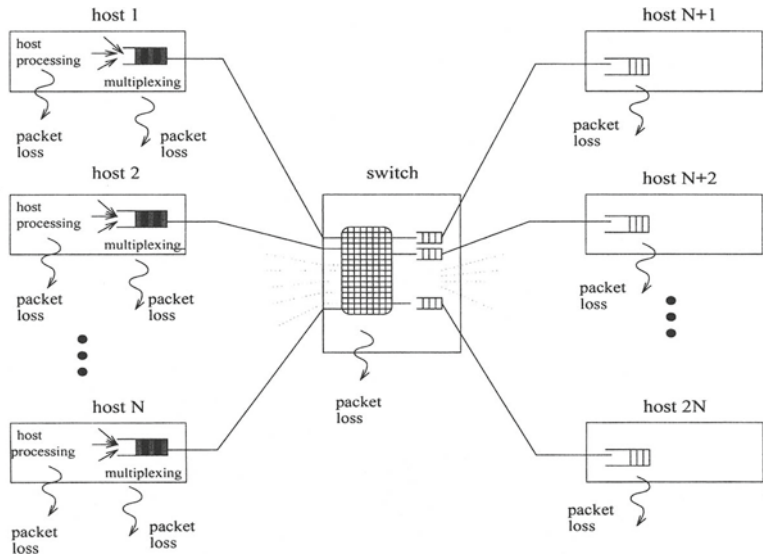


Figure 3 N senders to N receivers configuration

interval is the interval based upon the frame rate, e.g., 24 frames per second implies a fixed frame interval of $1/24 = 41.67$ milliseconds. In a jitter-free environment, all of the received intervals would be equal to the fixed frame interval. The **percentage of lost frames** is the percentage of lost frames during a single run, or averaged among several single runs. Any frame missing a packet is considered lost.

Data Collection. In both configurations (Figures 2 and 3), the sender(s) and receiver maintained logs of packet departures and arrivals. The sending host recorded the number of packets sent, and the average send time. If the sending host was transmitting N streams (at 24 frames per second) and the average send time was larger than $1/24 = 42.67$ milliseconds, this situation implies that the sending host could not support the transmission of N video streams at the 24 frame per second rate, i.e., the sending host was overloaded, and hence was introducing jitter which was not created by the network components. In these cases, the data was not used in this study.

On the receiving side, the receiving host receives the packets and demultiplexes them according to sender host id and port number. A log for each incoming video stream is kept. The number of the missing frames is recorded. Any frame missing a packet is considered lost. The time to receive each entire frame is recorded. The time between the receipt of an entire frame and the receipt of the next entire frame is recorded.

For all our experiments, segments were 'played' for 3 minutes and 40 seconds. The data was 'played' by reading a file which consisted of an enumeration of frames and their corresponding byte sizes. To ensure accuracy, individual runs were executed multiple times. When appropriate, replicated results were averaged and the standard deviation computed and reported.

Test	Fore ASX-200	Cisco Hyperswitch A10
7.488 Mbps CBR stream switched across one port	10.57 microsecs	39.90 microsecs
7.488 Mbps CBR stream plus bursty IP data to 10% capacity switched across the same port	27.19 microsecs	52.67 microsecs
7.488 Mbps CBR stream plus burst IP data to 50% capacity switched across the same port	77.89 microsecs	130.21 microsecs

Figure 4 Data collected at the ENL[16]: Latency per port under various loads.

3 EXPERIMENTS

3.1 Previous Related Work

A previous study [Mandeville (1995)] has shown the Fore ASX-200 to outperform the Cisco Hyperswitch in terms of latency and jitter (see Figure 4). This was not a surprising result considering the additional complexity of the Cisco Hyperswitch's buffering scheme. One purpose of this section is to see whether the additional functionality (credit-based input-output port buffering scheme), and hence overhead, as provided by the Cisco Hyperswitch, will serve to facilitate, or serve as an additional drawback, on the performance of VBR video transmission. Comparing the Fore Systems ASX-200 with the Cisco Hyperswitch performance is particularly illuminating since the ASX-200 provides only output buffering.

3.2 Performance Measurements

Experiment #1: JPEG, MPEG-1, MPEG-2. In this experiment, the test configuration used was the 4 sender to 1 receiver model. The protocol suite used was UDP AAL5 ATM. The average (among all streams in the same run) percentage of frames lost as a function of the number of streams injected by the sending host is shown in Figure 5. JPEG, MPEG-1, and MPEG-2 sequences from the Star Wars movies are compared. Despite the poorer quality JPEG video sequence and the lower capture rate of 15 frames/sec (compared to 24 frames/sec for the MPEG-1 sequence), the percentage of lost frames for the JPEG coded sequence was still higher than the percentage of lost frames for the MPEG-1 coded sequence (see Figure 5). The higher loss rates of the JPEG streams can be attributed to the following. The average frame size in the JPEG coded sequence consisted of 10944 Bytes. The average frame size in the MPEG-1 coded sequence consisted of 5184 Bytes. The average 'I' frame size in the MPEG-1 sequence was close to 11000 Bytes (which is close to the average frame size from the JPEG sequence). A sequence with an average larger frame size would result in greater losses because (i) any cell or packet loss would result in the entire frame being lost (i.e., smaller frames are less likely to be lost than larger frames), and (ii) a stream with a higher bit rate would cause more contention

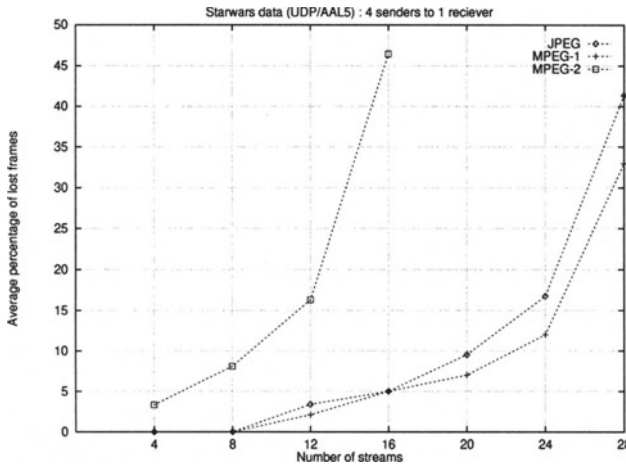


Figure 5 Experiment #1: Frame loss rates for JPEG, MPEG-1, and MPEG-2

throughout its transmission and hence packets from that stream would be more likely to be discarded. For the same reason, due to the much larger frames in the MPEG-2 stream, the average percentage of lost frames for the MPEG-2 sequence was the greatest.

For the remainder of our experiments, we used the MPEG-1 sequence from the Star Wars movie. We chose to use the MPEG-1 sequence rather than the JPEG sequence because it is the most likely type of coded video to be used in the future; MPEG-1 provides higher compression ratios and lower mean bit rates than JPEG. The MPEG-2 stream data bit rate was too high (i.e., caused too many lost frames) to produce meaningful results in our particular environment. Thus, hereafter, the term MPEG implies MPEG-1.

As mentioned before, we used the TCP/IP and UDP/IP protocols as the transport mechanisms for transporting the video streams.

‘Acceptable’ visual quality. Continuous media traffic, such as coded video, have the real-time requirement that frames be displayed sequentially (continuously) with no prolonged delays between any pair of consecutive frames; the interarrival time, or jitter, between frames must be bounded.

When network congestion occurs, frames may be discarded (via buffer overflow), or in the case of TCP, may be discarded and then re-transmitted. If a frame arrives late, it will cause jerkiness in the visual medium. If a frame never arrives (i.e., is lost) its absence will also cause jerkiness in the visual medium. In MPEG, which consists of ‘I’, ‘P’, and ‘B’ frames, some frames are more important than others [LeGall (1991), Tsang (1996)]. ‘I’ frames are complete bit images. They must be received regularly in order to re-generate high quality images. ‘P’ and ‘B’ frames are used to ‘refresh’ the current image. If a ‘P’ or ‘B’ frame is lost, the decoder may be able to ‘guess’, or estimate, the lost frame until the next ‘I’ frame arrives. ‘I’ frames serve as a reference point for creating ‘B’ and ‘P’

number of streams	Fore ASX-200				Cisco Hyperswitch			
	max	min	avg	std dev	max	min	avg	std dev
4 senders to 1 receiver								
32 streams (8 / sender)	94.10%	34.92%	72.25%	15.3	79.35%	2.32%	45.18%	11.0
28 streams (7 / sender)	75.02%	1.71%	32.71%	12.5	58.10%	2.49%	22.50%	9.1
24 streams (6 / streams)	33.83%	1.60%	11.81%	10.2	21.42%	1.46%	5.00%	2.6
20 streams (5 / sender)	24.80%	0.93%	6.74%	3.6	5.45%	0.32%	1.95%	0.8
16 streams (4 / sender)	9.35%	0.74%	5.32%	2.3	1.49%	0.26%	0.98%	0.5
2 senders to 1 receiver								
28 streams (14 / sender)	56.9%	1.42%	21.32%	11.6	42.9%	1.48%	18.21%	7.3
24 streams (12 / sender)	21.0%	1.40%	9.36%	5.6	14.1%	1.68%	4.63%	1.5
20 streams (10 / sender)	16.01%	0.73%	6.87%	4.8	3.6%	0.33%	1.99%	1.3
16 streams (8 / sender)	5.43%	0.25%	3.79%	1.6	1.12%	0.0%	1.02%	0.4

Figure 6 Experiment #2: N senders to 1 receiver: frame loss comparisons for dif ferent switches

frames. Hence, ‘I’ frames are more important to maintaining high visual quality; their loss is much more apparent to a viewer.

In our experiments, we define a loss of more than 10% to be visually noticeable to most viewers and hence unacceptable. The worst case occurs when all 10% of the lost frames are ‘I’ frames. At 24 frames per second, a 10% loss translates into a 21.6 frame per second frame rate. This is in the range of what is usually judged to be ‘acceptable’ visual quality.

Experiment #2: Output Port Contention Measurements In this experiment, the test configuration used was the N sender to 1 receiver model (see Figure 2). The protocol suite was UDP AAL5 ATM.

Figure 6 depicts the two test scenarios. In the first scenario, we transmitted multiple video streams from **4 sending hosts to 1 receiving host**. In the second scenario, we transmitted multiple video streams from **2 sending hosts to 1 receiving host**.

This test was performed in order to measure how well each switch handled contention

number of streams	Fore ASX-200				Cisco Hyperswitch			
	max	min	avg	std dev	max	min	avg	std dev
128 streams (32 sender)	92.92 %	48.1%	73.46 %	12.1	76.91 %	2.19 %	40.25 %	14.2
112 streams (28 sender)	70.81 %	36.3%	53.32 %	9.9	68.89 %	3.14 %	23.82 %	11.3
96 streams (24 sender)	22.26%	1.8%	10.08%	5.0	23.32%	0.7%	9.69%	4.1
80 streams (20 sender)	13.16%	1.0%	2.55%	1.8	3.18%	0.4%	1.15%	0.4

Figure 7 Experiment #3: N senders to N receivers: frame loss comparisons for different switches

at an output port. Examining Figure 6, the first item to note is that given the same number of streams, for both the Fore and Cisco switch, the **2 senders to 1 receiver** case has a consistently lower frame loss rate than the **4 senders to 1 receiver** case. Given the same number of streams, while only varying the number of senders, implies that for a greater number of senders there is more contention at the output port and hence greater frame loss; for a fewer number of senders, more streams are already superimposed (multiplexed) at each sending host and less at the output port of the switch.

The comparison between how well the Cisco A100 Hyperswitch and Fore Systems ASX-200 switch handled output contention shows that the Cisco switch is able to consistently lose fewer frames than the Fore ASX-200 switch. Again, this may be attributed to there being less contention at the output port of the Cisco switch than the Fore switch. Recall, that the Cisco switch uses an input-output credit-based buffering scheme where loss never occurs at the output ports; data is only sent to an output port if that output port has sent a credit to the input port, where the data is being held, signaling that it has a certain amount of empty buffer space. Thus in the Cisco switch, contention is distributed among the input ports whereas in the Fore switch (for the N sender to 1 receiver case) all the contention occurs at a single output port.

Experiment #3: Cross Traffic Measurements. In this experiment, the test configuration used was the N sender to N receiver model (see Figure 3). The protocol suite was UDP AAL5 ATM.

In this experiment, we transmitted multiple video streams between disjoint sets of 4 sending hosts and 4 receiving hosts, i.e., 4 pairs of sending-receiving hosts. Each pair of hosts supported the transmission of multiple MPEG streams. Figure 7 depicts the frame loss percentages as a function of load for both the Cisco and Fore switches. It can be seen that the switch fabric of the Cisco switch is able to handle the same traffic (as was handled by the Fore switch) but with a consistently lower percentage of lost frames. Again this may be attributed to the more sophisticated buffering scheme provided by the Cisco Hyperswitch. In the Cisco switch, contention is distributed among the input ports first

number of streams	Fore ASX-200			Cisco Hyperswitch		
	[2, 4)	[4, 6)	[6, infinity)	[2, 4)	[4, 6)	[6, infinity)
32 streams (8 / sender)	21%	13%	7%	17%	12%	5%
28 streams (7 / sender)	17%	6%	3%	14%	7%	2%
24 streams (6 / sender)	15%	4%	3%	12%	5%	0%
20 streams (5 / sender)	10%	3%	0%	6%	1%	0%

Figure 8 Experiment #4: 4 senders to 1 receiver - percentage delayed

whereas in the Fore switch all the contention occurs at the switch fabric (TDM bus) immediately.

Experiment #4: Jitter vs. number of multiplexed streams. This experiment was performed using TCP on ATM AAL5.

Figure 8 depicts the percentages of frames delayed for varying numbers of fixed frame intervals. Recall, that loss is not a performance measure in this experiment since TCP rarely loses packets. Packets which are delayed between x fixed frame intervals and $y - 1$ fixed frame intervals are said to be delayed by $[x, y)$, i.e., their received interval is between $[x, y)$.

Using TCP, packets which are discarded by the network are re-transmitted. Thus packets (and hence frames) are very rarely lost. Packets are discarded only after a pre-set number of re-transmissions occur. However, this situation occurs very rarely, and we did not observe any loss for any of the TCP experiments. We assumed that the receiving host could buffer up to 3 fixed frame intervals (or 3 frames). This is a reasonable assumption because 'B' frames may reference either forward or backward frames; both the last and next 'P' or 'I' frames must be transmitted before the 'B' frames may be sent. Thus the receiver host must be able to buffer a minimal of 3 frames. Hence a frame received within 3 fixed frame intervals will not cause jitter to occur. A frame which arrives outside of 3 fixed frame intervals is considered lost (undisplayable) and may contribute to the overall perceived jitter. Thus, similarly to what is acceptable for lost frames, we define an intolerable number of delays (i.e., losses) to be 10%; if more than 10% of the frames are delayed (beyond 3 fixed frame intervals), the visual quality is unacceptable.

Assuming frame delays less than or equal to 3 fixed frame intervals are tolerable (due to buffering), TCP can support up to (28, Fore switch) and (30, Cisco switch) multiplexed streams with only 9% of the received intervals exceeding 3 fixed frame intervals.

Note that TCP is able to support a far greater number of multiplexed streams (within reasonable inter-arrival delay constraints) than UDP (within reasonable frame loss rates). Also for TCP the variation in jitter between individual multiplexed streams from the

same run was found to be negligible. Recall that for UDP the variation in frame loss rates between individual multiplexed streams from the same run was very high.

4 DISCUSSION OF RESULTS

Recently many new distributed multimedia applications have been proposed. These new applications are based upon the transmission of digital information such as coded video, voice, high-resolution images, data, and graphics. For the network provider, the challenge lies in providing adequate network support for these various data types in order to operate on a communications infrastructure with a shared switching and transmission facility.

From our results, we conclude with the following points of discussion.

Fast packet switching? Much emphasis has been placed on the need for fast packet switches as a means of enabling the high transport speeds of ATM [Biagioni (1993), Fan (1994)]. This need for fast packet switches has brought about the philosophy that switches with the simplest, yet efficient, architectures are the most desirable. The *ideal* architecture which has been cited is one which provides either no input buffering (or an input buffering scheme which does not have the ‘head-of-line’ blocking problem) and buffers at each output port [Biagioni (1993), Mandeville (1995)]. This appears to have been the philosophy behind the switch architecture of the Fore Systems ATM switches. And, indeed, the Fore Systems switch does provide lower jitter and delay than most other available ATM switches including the Cisco Hyperswitch. This was shown in [Mandeville (1995)].

How do a few microsecond jitter differences, or even a few hundred microsecond jitter differences translate into the application Quality of Service (QoS) for video transmission, i.e., frame inter-arrival time of 30-40 milliseconds? From the results in our experiments, we can infer that there is not a direct translation and (within reasonable bounds) the jitter (latency) differences and end-to-end performance of video need not even be related.

From our experimental results, as well as the results in [Mandeville (1995)], we reason that the Cisco switch performed superiorly because it uses a more sophisticated buffering scheme (at input and output ports) which uses a form of flow control to ‘smooth’ the traffic between the input and output ports [Fan (1994)]; fewer cells are lost due to buffer overflow. The Fore switch uses pure output buffering so all contention must be resolved at the output buffer - hence a greater number of buffer overflows may occur. So in the end, it did not matter that the Cisco switch incurred more latency (jitter) than the Fore switch!

Network Transport Protocols. Due to the real-time nature of video traffic, a guaranteed service protocol, or a best-effort service protocol with real-time scheduling at individual switch nodes, has usually been suggested as the most appropriate type of protocol for multimedia traffic such as video. Protocols which provide end-to-end flow control and re-transmissions, such as TCP, have been considered unsuitable because it has usually been thought that the delays caused would result in packets missing their bounded delay requirements and hence becoming meaningless (discarded). From our experiments, we observed TCP’s sliding window protocol to have the effect of smoothing traffic burstiness, as well as still being able to deliver packets within their deadlines. Thus our results

showed that the appropriate type of flow control may greatly improve application level performance significantly over a best-effort transport mechanism. In our particular environment TCP worked very well. This is not to suggest TCP as an appropriate video transport layer. Actually, in other environments (e.g., a wide-area network), it is very unlikely TCP would perform as well. The important point is that explicit flow control, either or a combination of link-link or end-end mechanisms, must be used to improve the performance of real-time traffic such as video. Best-effort delivery, in conjunction with real-time scheduling at individual switch nodes, which has been widely suggested as the most appropriate transport mechanism for real-time traffic such as video, is unlikely to be adequate.

Traffic Control. Traffic control for VBR video traffic is difficult to implement due to the real-time nature of the traffic. But, we believe it to be a necessary measure to increase network utilization and provide more stringent guarantees to continuous media traffic.

When the hosts and the network are stressed, the experimental data show that the burstiness of the variable bit rate coded video streams is a significant factor in the resulting performance degradation. We saw that controlling burstiness (through end-end flow control or switch buffering techniques) results in significantly less packet losses. This implies a fewer number of re-transmissions for transport protocols which guarantee reliability through re-transmissions, such as TCP, as well as for unreliable transport protocols such as UDP. Losses are significant when transmitting traffic types with large pre-specified (non-changeable) frame sizes like coded video. If any packet of a frame is lost, the entire frame may be discarded. Thus a relatively small percentage of lost ATM cells may translate into a relatively large frame loss percentage in data traffic (such as coded video) which must use large frame sizes. Our results show that traffic control results in a significantly decreased frame loss rate while maintaining acceptable jitter and loss bounds.

REFERENCES

- Biagioni, E., Coope, E., Samsom, R. (1993) Designing a Practical ATM LAN. *IEEE Network*, March 1993.
- Cole, B. (1993) The Technology Framework. *IEEE Spectrum*, March 1993.
- Fan, R., Suzuki, H., Yamada, K., Matsuura, N. (1994) Expandable ATM Switch Architecture (XATOM) for ATM LANs. *CISCO Systems Journal*.
- Heffes, H., Lucantoni, D. (1986) A Markov Modulated Characterization of Packetized Voice and Data Traffic and Related Statistical Multiplexer Performance. *IEEE Journal on Selected Areas of Communication*, September 1986.
- IBM (1994) TURBOWAYS 100 ATM Adapter: User's Guide. March 1994.
- Le Gall, D. (1991) MPEG: A Video Compression Standard for Multimedia Applications. *Communications of the ACM*, April 1991.
- Lyles, J., Swinehart, D. (1992) The Emerging Gigabit Environment and the Role of Local ATM. *IEEE Communications Magazine*, April 1992.
- Maglaris, B., Anastassiou, D., Sen, P., Karlsson, G., Robbins, J. (1988) Performance Models of Statistical Multiplexing in Packet Video Communications. *IEEE Transactions on Communications*, July 1988.
- Mandeville, R. (1995) The ATM Stress Test: Which Switches Survived ? *Data Commu-*

- nication, March 1995.
- Norros, I., Roberts, J., Simonian, A., Virtamo, J. (1991) The Superposition of Variable Bit Rate Sources in an ATM Multiplexer. *IEEE Journal on Selected Areas in Communications*, April 1991.
- Pancha, P., El Zarki, M. (1994) MPEG Coding for Variable Bit Rate Video Transmission. *IEEE Communications Magazine*, May 1994.
- Tsang, R., Pavan, A., Du, D. (1996) Experiments with Video Transmission over an Asynchronous Transfer Network. *ACM/Springer-Verlag Multimedia Systems Journal*, August 1996.
- Wallace, G. (1991) The JPEG Still Picture Compression Standard. *Communications of the ACM*, April 1991.

Throughput optimization for multimedia applications over high speed networks

S. Zeadally G. Gheorghiu A.F.J. Levi
Department of Electrical Engineering
University of Southern California
University Park, Los Angeles, California 90089, USA
Phone: (213)740-1450
Fax: (213)740-9280
E-mail: zeadally@marco.usc.edu

Abstract

Digital video services, scientific visualization and other multimedia applications require delivery of high network throughput to end user applications. In this paper we identify some of the bottlenecks in the data path between high-speed networks and applications which are responsible for poor application performance. We then present solutions to overcome these bottlenecks at various levels namely: network, operating system, and user. Finally, we show the effectiveness of these solutions on the performance of our multimedia applications.

Keywords

Multimedia, networking, operating systems, performance, TCP/IP

1. INTRODUCTION

The past few years have seen development of applications with high bandwidth requirements such as medical image transfer, video conferencing, scientific process simulation, and visualization. Popular local area networks such as Ethernet and Token Ring (4 Mbits/s and 16 Mbits/s) are incapable of providing the bandwidth needed by these multimedia applications. New network technologies such as Fiber Distributed Data Interface (FDDI), Fast Ethernet (100BASE-T), and Asynchronous Transfer Mode (ATM) have emerged and are capable of providing high bandwidth to users' desktops. However, the challenge remains for operating system designers and application developers to deliver the bandwidth of these networks to end user applications. In order for applications to reap the benefits of high-speed networks, the entire path from network to application must be optimized. This involves removing bottlenecks introduced both at the operating system and application levels as depicted in Figure 1.

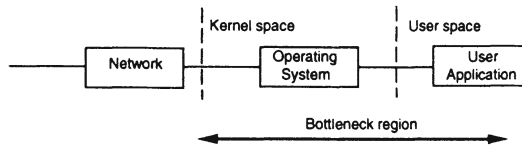


Figure 1 Illustration of bottlenecks in typical multimedia network-application data path for conventional end-systems considered in this study.

This paper describes our experiences delivering multimedia services over high-speed networks and our attempts to optimize network-application throughput. We deal with *real-life multimedia applications* as opposed to using *raw data*. The use of raw data in performance tests provides an upper bound on achievable performance that can be delivered by the underlying software and hardware. However, raw data does not identify other bottlenecks that are normally associated with actual applications such as the presentation of information (e.g. video display in a window) or the effect of running multiple applications concurrently. Some of the work done by other researchers in similar areas includes (Keller, 1993) (Patel, 1993) (Rowe, 1993). They have focused principally on optimizing only one area in the network-application data path (e.g. at application level). Our work differs from these efforts in that we apply optimizations wherever possible to the *entire* data path between network and applications as opposed to focusing on just one stage in the transfer whereby later stages introduce other overheads which degrade the overall application throughput. For instance, no matter how fast the network delivers data to the host, if the operating system itself cannot transfer the data to the application at a high enough sustained rate, then there will be a bottleneck introduced in the network-application data path.

The structure of this paper is organized as follows. In section 2, we discuss techniques that we applied to increase application-application throughput. Section 3 describes the experimental setup we have used for our multimedia applications. Section 4 presents an analysis of in-host data movement for our scientific graphics visualization application and digital video playback. Section 5 discusses the results obtained by implementing the various optimization techniques given in section 2. In section 6, we analyze the overall performance when multiple applications are running. Finally, section 7 makes some concluding remarks and presents future work.

2. IMPROVING APPLICATION-APPLICATION THROUGHPUT

To achieve high application-application throughput in a network environment, it is essential that high sustained throughput be delivered by the network, operating system, and application.

Recent advances in the performance of network physical layers have essentially solved the network bandwidth problem. As a result, it is now feasible to deliver large volumes of data at high rates with minimal loss. In this work, we have used a conventional Ethernet network, a higher speed FDDI network, and an experimental network called Jetstream (Watson, 94) (section 3) in order to study their impact on application performance. The basic physical layer characteristics of these networks include: Manchester code Ethernet signalling at 20 Mbits/s giving a maximum data rate of 10 Mbits/s, 4b/5b coding on FDDI signalling at 125 Mbits/s delivering a maximum data rate of 100 Mbits/s, and 16b/20b coding for Jetstream signalling at 1 Gbit/s giving a maximum data rate of 800 Mbits/s. These maximum data rates are reduced by media access control, network subsystem, application, and of course, limitations imposed by the host architecture.

The last few years have witnessed significant hardware improvements that have led to the development of powerful computers. Some of these improvements include: increased CPU performance, high bus bandwidth, large memories, and fast disk systems. However, there has been little change in the structure of conventional operating systems such as UNIX, consequently the availability of new hardware technologies has not been exploited to the fullest. This has made existing operating systems become a bottleneck in end systems. Well-known overheads include data copying,

network protocol processing, context switches, and interrupts (Kanakia, 1988) (Pasquale, 1992) (Ousterhout, 1990). Several techniques have been proposed and implemented to avoid these overheads (Druschel, 1993) (Dittia, 1995) (Jacobson, 1990) (Pasquale, 1994). In this work, we minimized data copying by using a UNIX kernel which supports 'single-copy' TCP/IP, a modified version of TCP/IP. Throughout this paper, we refer to 'two-copy' TCP/IP as the standard version that normally comes with UNIX operating systems. In this case, data transfer between network and application normally involves two copies: the first copy is between a network buffer and kernel buffer followed by a copy from the kernel buffer to a user application (for an incoming packet). The reverse takes place for an outgoing packet. However, in the case of a single-copy TCP/IP implementation we use for our experiments in this paper, there is only one data copy between network and application thereby eliminating the copy to kernel buffer. Moreover, the single-copy implementation of TCP also supports RFC 1323 window scaling (Jacobson, 1992), and is capable of calculating checksum during data movement.

It is not easy to come up with general techniques to increase throughput at the application level. The main reason is that different applications have different requirements and each is implemented in its own way. However, it is true that multimedia applications have a common element: they all present information (e.g. video display) to the end user. Most desktop applications running on UNIX platforms are built on standard X window systems to increase their ease of use, and offer a common look and feel to users. The X window system has become the de facto standard graphical user interface for UNIX systems. We argue that there is scope for improving application performance in the X environment in the area of data presentation. In this context, we note that without careful tuning, data display by the X server can degrade performance in the final delivery of information to the user. Our choice was to use the X shared memory extensions (Corbet, 1991) in order to speed up image display.

The usual way to display an image is to use the X11 library call *XPutImage()* on an application's data. The call to *XPutImage()* moves data from the application's buffer via Inter-Process Communication (IPC) to a private buffer of the X server (using UNIX domain sockets when the X client and the X server are on the same machine). The data is then moved by the X server to the frame buffer. With X shared memory extensions support, there is no data movement involved between the application and the X server. Instead the image data is placed into a memory segment that is shared between the application and the X server. In this case, a call to *XShmPutImage()* allows the X server to move data directly from the shared memory segment containing the application image data to the frame buffer.

3. EXPERIMENTAL ARRANGEMENT FOR MULTIMEDIA APPLICATIONS

The experiments described in this section have been carried out between two HP 9000 Series 700 workstations (99 MHz PA-RISC) which reside on the Jetstream network. Jetstream is a Gbit/s token-ring network which uses copper coaxial or fibre optic cable for the physical link. The network adapter for the HP 9000 Series 700 workstations is made up of two cards - one is called Afterburner which is equipped with 1 MByte of video random access memory used in a dual ported configuration. Afterburner is the host interface; the other card, Jetstream, is the link adapter. The shared memory present on the Afterburner board enables the support of *single-copy* implementations of network protocols such as TCP/IP and UDP/IP. Further details of Afterburner and Jetstream are given in (Watson, 1994) and (Dalton, 1993).

The hardware architecture of the host is illustrated in Figure 3. The Standard Graphics Connector (SGC) (DeBaets, 1992) is the system bus used on the HP 9000 Series 700 workstation and has a maximum data transfer rate of a Gbit/s. However, it is only possible to achieve a maximum of 400 Mbits/s transfer rate by the CPU between memory and input/output (I/O) space (e.g. graphics devices or network interface) (Frink, 1992). A principal feature of the hardware architecture is the memory and system bus controller chip which connects the CPU to memory and the I/O system components. The system bus controller chip communicates to the SGC bus via two system bus interface chips. The workstations used in our experiments have 128 MByte of main memory and two GByte of hard disk. The operating system used was HP-UX 9.01 with single-copy TCP/IP support.



Figure 2 Visualization application.

We have used two different applications in our experiments: one is a scientific visualization application and the other is digital video playback.

The visualization application uses a series of gray scale images (8 bit/pixel) of a volume rendered CAT-scan medical image of a child head as shown in Figure 2. The head can be rotated and viewed at different angles. In our experiment, the images are stored on a remote server and sent over the network to the client machine which displays the images. The image set consists of 20 image frames each of size 512x512 pixels (almost 2.1 Mbits per frame) and stored in pixmap format. This makes direct display by the X server possible without requiring any further manipulation. The user interface to the visualization application supports simple operations such as play, stop, and rewind.

The video application uses the PowerVideo700 hardware video codec (compression / decompression) from Parallax (Parallax, 1994) which supports motion-JPEG. Motion-JPEG applies JPEG (Joint Photographic Experts Group, a standardized image compression technique for still images) to individual frames of a video sequence. The video board is capable of handling high quality video at 30 frames per second in real-time during either recording or playback sessions. The PowerVideo700 is an overlay card which resides in one of the EISA slots of the EISA interface attached to the SGC bus as shown in Figure 2. We have used the MovieTool software from Parallax for recording and playing digital video stored as motion JPEG files.

For our experiments, the files used during video playback were stored on the hard disk of a remote machine. This disk was mounted on the host machine using NFS via the Jetstream interface (details are given in section 4). In a typical video playback session, the use of NFS enables the host machine to receive compressed video clips over the Jetstream network. The video board decompresses the incoming compressed video stream. The analog signals originating from the graphics card are digitized and the result is overlaid with the uncompressed video image. After the overlay is completed, the entire frame is converted back to analog and sent to the monitor (Figure 3).

In all measurement tests for both the video and scientific visualization applications, we use average playback frame rate as our quantitative metric to characterize application performance. Moreover, in all experiments, these applications were run as normal user processes, along with the usual system processes and daemons in the background.

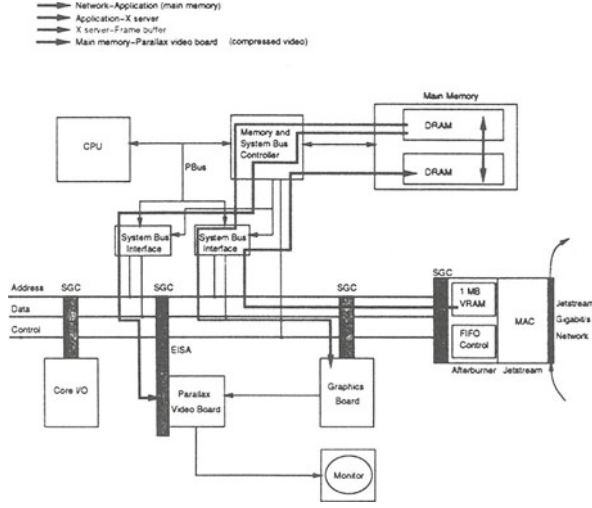


Figure 3 HP 9000 Series 700 architecture data paths for applications displaying network data and video playback. A network adapter connects the workstation to the Jetstream network. A Parallax video board performs video decompression and drives the monitor. Graphics data is passed from main memory to the graphics board and overlaid on the monitor by the Parallax video board.

4. ANALYSIS OF IN-HOST DATA MOVEMENT

In this section, we identify the data paths used when running the video and visualization applications based on the architecture presented in Figure 3.

To understand the impact of the underlying architecture on application performance, we measured the throughput at different stages when moving data from the network to the X window display for the visualization application. The major aim of performing such analysis is to identify areas where performance can be improved, and at the same time assess the suitability of the HP 9000 Series 700 workstation architecture in supporting high-speed network applications. Our observations are applicable to other similar networked multimedia applications (e.g. medical imaging, video conferencing).

Equation 1 summarizes the inverse of total throughput R for typical applications that read data from the network and use the X window system for display:

$$\frac{1}{R} = \frac{1}{Z_{\text{Network-Application}}} + \frac{1}{Z_{\text{Application-Xserver}}} + \frac{1}{Z_{\text{Xserver-Framebuffer}}} \quad (\text{Equation 1}),$$

where Z represents the throughput at different key stages of the data path from network to frame buffer as depicted in Figure 3. Furthermore, Equation 1 applies to conventional bus-based systems of the type used in our experiments.

To verify the correctness of the above equation, we ran the graphics visualization application via Jetstream and obtained a frame rate of 33 frames/second. This corresponds to a throughput of 69.3 Mbits/s (33 frames per second

multiplied by 2.1 Mbits per frame). We then measured the throughput values corresponding to the three stages of the data path of Equation 1 as follows:

- The application reads data from a socket into its buffer. The fact that we are using a single-copy TCP/IP stack allows direct data transfer from a network interface buffer to the application's buffer (avoiding the additional copy to a kernel buffer). The throughput obtained was 200 Mbits/s ($Z_{Network-Application}$).
- The application acting as an X client sends the data to the X server using inter-process communication. The rate of data movement is 240 Mbits/s ($Z_{Application-Xserver}$).
- The X server then moves the data to the frame buffer. The transfer rate obtained along this path is 230 Mbits/s ($Z_{Xserver-Framebuffer}$).

Using the measured throughput values, we calculated the total throughput R from Equation 1 and obtained 74 Mbits/s. This value is slightly higher than the observed Jetstream throughput of 69.3 Mbits/s. The difference of 4.7 Mbits/s is due to the fact that we did not take into account various overheads such as system calls, context switches, interrupts and memory allocation by the X server during data copying.

To understand the impact of running multiple network applications on the performance of the end system, we have chosen to simultaneously run both the video and the visualization applications over Jetstream. For this to be possible in the case of the video application, we used NFS over the Jetstream network interface as shown in Figure 4. This enables playback of digital video clips stored on a remote disk which has been mounted on the host machine (used as an NFS client). Our UNIX kernel supports NFS 2.0 which uses UDP.

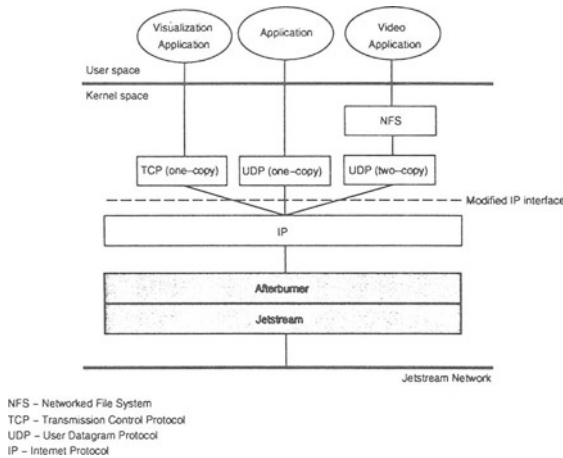


Figure 4 Protocol stacks with one-copy and two-copy support. IP layer has been modified to allow NFS support (which uses two-copy UDP/IP) to co-exist with applications using single-copy UDP/IP.

Although the UNIX kernel we have used does support both single-copy TCP/IP and single-copy UDP/IP stacks, it was not possible for NFS to use the single-copy UDP/IP stack. This is because NFS does not understand the buffer structures used in the single-copy implementation of UDP/IP. We did not consider it worth modifying NFS to allow it to support our single-copy protocol stacks. The justification was that we do not see significant throughput improvement with a version of NFS that allows single-copy protocols since the disk (at the NFS server) will still be

the bottleneck (although latency would be slightly better). For our experiments, it would have been sufficient to use a single-copy TCP/IP stack (for the visualization application) and a two-copy UDP/IP stack (for video application over NFS) by simply disabling single-copy support for UDP/IP in the UNIX kernel. However, the disadvantage of this approach is that it prevents other applications from using single-copy UDP/IP. Our solution was to modify the IP layer in order to distinguish packets destined for NFS which will use two-copy UDP/IP from all other incoming network packets which will use single-copy UDP/IP or single-copy TCP/IP (Figure 4).

5. NETWORK MEASUREMENTS AND RESULTS

Initial experiments were conducted using *raw data* to investigate how much of the available network bandwidth can actually be delivered to the application. Figure 5 presents the observed throughput using a two-copy TCP/IP stack over Ethernet, FDDI and Jetstream. The laboratory Ethernet has been used for Ethernet tests. In the case of FDDI, an EISA FDDI adapter designed for HP 9000/700 EISA systems was used to attach to an FDDI network. Performance measurements were made with a tool called *netperf* (Jones, 1993) which measures the transfer of data from a producer process (generating the data) to a consumer (receiving data) running on a remote machine. A socket buffer size of 32 KBytes (KB) has been used at both workstations. We have chosen this socket buffer size because it is close to the limit possible on standard UNIX systems (usually 48 KB) and we wanted to demonstrate the throughput achievable with an unmodified kernel. The maximum network data bandwidths for Ethernet, FDDI, and Jetstream are 10 Mbits/s, 100 Mbits/s, and 800 Mbits/s respectively. However, the maximum raw data throughput values obtained using two-copy TCP/IP in our experiments were around 9.6 Mbits/s, 74 Mbits/s and 90 Mbits/s for Ethernet, FDDI, and Jetstream respectively. These results confirm our initial assumption that the end system has become the bottleneck in a high-speed network environment.

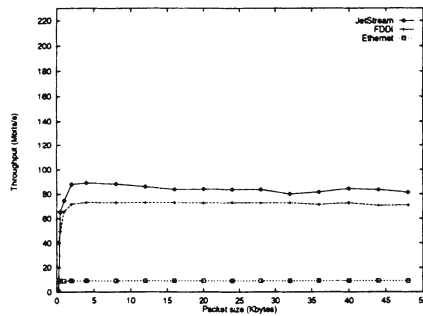


Figure 5: Measured raw data throughput over Ethernet, FDDI and Jetstream using two-copy TCP/IP and 32 KB socket buffer size.

Figure 6 shows raw data application throughput using single-copy TCP/IP over the Jetstream network. The use of a single-copy kernel increases throughput by almost 56% using 32 KB socket buffer size. Increasing the socket buffer size, if the kernel allows it, results in higher throughput values. For instance, the single-copy kernel enabled us to specify a socket buffer size of 256 KB. In this case the maximum throughput achieved was around 200 Mbits/s.

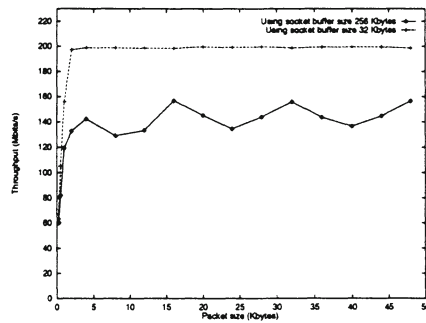


Figure 6 Measured raw data throughput over Jetstream using single-copy TCP/IP and socket buffer sizes of 32 KB and 256 KB.

Having established the achievable throughput possible using raw data, we then used the visualization application to measure the maximum throughput for a real application running over various networks. In contrast to raw data throughput results, we obtained frame rates and corresponding throughput values given in Table 1. Two important observations can be made: first, performance based on raw data is not enough to characterize application throughput; second, in the case of slow networks like Ethernet, software (e.g. operating system) on end systems is able to deliver most of the available network bandwidth to the application. However, as network speed increases, the discrepancy between network bandwidth and actual application throughput is increasing.

Table 1 Measured graphics visualization application frame rate and the equivalent throughput using two-copy TCP/IP and 32 KB socket buffer size over Ethernet, FDDI and Jetstream

Network type	Frame rate (frames/s)	Throughput (Mbits/s)
Ethernet	4	8.4
FDDI	11	23.1
Jetstream	22	46.2

We now discuss how the various optimizations mentioned in previous sections can be applied to the data path described by the three terms of Equation 1.

- **Network:** We use the Jetstream Gbit/s network capable of supporting high bandwidth applications.
- **Operating system:** A UNIX kernel that supports single-copy TCP/IP has been used. This allows direct data copy from the network to the application. As a result, data movement, considered to be the major bottleneck in current operating systems, is minimized. Moreover, network protocol overheads such as checksum calculations have also been significantly reduced. We have therefore optimized the first term of Equation 1 with $Z_{\text{Network-Application}}$ being 200 Mbits/s. This value is the maximum throughput obtained when using raw data as illustrated in Figure 6.
- **Application:** We have used the X shared memory extensions to eliminate data movement from the application to the X server. This optimizes the overall throughput by eliminating the second term $Z_{\text{Application-Xserver}}$ of Equation 1. It is also worth noting that applications should exploit the capability of using large socket buffer size

whenever the kernel allows it. Although this is done at application level, it influences the throughput between network and application at the operating system level.

Figure 7 summarizes the effects of the various optimization approaches on the performance of the visualization application. It is interesting to note from the graph that for socket buffer sizes up to 48 KB (the maximum allowable by standard UNIX kernels), the application performs better using two-copy TCP/IP and X shared memory extensions than using single-copy TCP/IP without X shared memory extensions.

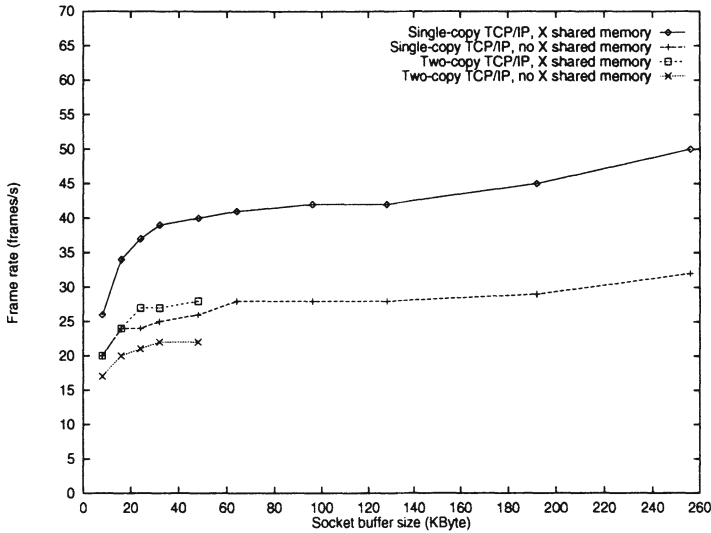


Figure 7 Measured effect of single-copy TCP/IP, X shared memory extensions, and socket buffer size on final throughput for the graphics visualization application.

The underlying architecture did not allow further optimization of the last term of Equation 1. This is because the system and memory bus controller shown in Figure 3 becomes the bottleneck when subjected to intensive data traffic to and from main memory and system bus. This is not a problem for one way traffic between main memory and a peripheral device or vice-versa. However, it becomes a limitation in the case of networked multimedia applications where data flows in continuously from network to main memory and back out from main memory to graphics display. A possible solution is to transfer data directly from a network device to the frame buffer over the system bus, a mechanism commonly referred as *kernel-level streaming* (Murphy, 1996) (Fall, 1993). Unfortunately, the SGC bus implementation in the machines used for our experiments does not allow the needed slave-slave bus transactions.

After applying all the above optimizations, Equation 1 becomes:

$$\frac{1}{R} = \frac{1}{Z_{\text{Network-Application}}} + \frac{1}{Z_{\text{Xserver-Framebuffer}}} \quad (\text{Equation 2}).$$

Calculating the value of R from Equation 2, using 200 Mbits/s for $Z_{\text{Network-Application}}$ and 230 Mbits/s for $Z_{\text{Xserver-Framebuffer}}$, we obtain an overall expected throughput of 107 Mbits/s. To verify the correctness of Equation 2, we measured the average frame rate for the visualization application after we implemented all the above optimizations. The value obtained was 50 frames/second which translates to a throughput of 105 Mbits/s. The

difference between the measured and the expected values is smaller (2 Mbits/s) than that obtained in section 4 (i.e. 4.7 Mbits/s). This is because the elimination of the second term of Equation 1 has also reduced overheads such as memory allocation by the X server.

6. OVERALL PERFORMANCE FOR MULTIPLE APPLICATIONS

To quantify the impact of running *both* video and visualization applications on overall performance, we used the following metrics: CPU usage, frame rate (visualization application), and percentage of frames dropped (video application). CPU usage was measured using *Glance* (Hewlett-Packard, 1992), a performance monitor tool that comes with standard HP-UX operating system. For the video application, the number of frames dropped was obtained from the diagnostics information generated by the device driver of the Parallax video board (Parallax, 1994). Table 2 summarizes the results obtained using the Jetstream network. All tests have been performed on a kernel that supports single-copy TCP/IP. For the visualization application, we used a socket buffer size of 256 KB. Digital video playback was via NFS at 30 frames/second, the size of each frame being 512x380 pixels and 24 bit color per pixel.

To better understand the degradation in performance when both applications are running concurrently, we first make some observations on their performance when executed on their own. When the video application is running by itself, there are no video frames dropped. For the visualization application, the frame rate without using X shared memory is 33 frames/second. However, when running both applications, there was a 39% drop of frames displayed with video and the frame rate observed for visualization decreased to 26 frames/second as shown in the Table 2. The degradation of video performance is due to the high percentage of CPU time (56%) spent in system mode. This is because the visualization application uses IPC to move data to the X server which involves multiple kernel-user interactions. The 34% of CPU cycles left for user mode are not sufficient for the demands of video, which requires 42% user-mode CPU time. On the other hand, 56% of CPU time spent in system mode is not sufficient for the needs of the visualization application which requires the CPU to spend 66% of its time in system mode. Thus, it is evident that the conflicting requirements of the two applications affect their overall performance.

Table 2 Measurement of impact of CPU utilization on video and graphics visualization applications performance.

	<i>%CPU in user mode</i>	<i>%CPU in system mode</i>	<i>Frame rate (frames/s)</i>	<i>% frames dropped</i>
VIDEO	42	48	--	0
GRAPHICS	27	66	33	--
VIDEO & GRAPHICS	34	56	26	39
GRAPHICS-SHMEM	40	54	50	--
VIDEO & GRAPHICS-SHMEM	41	50	35	10
VIDEO	video playback application			
GRAPHICS	visualization application without X shared memory extensions			
GRAPHICS-SHMEM	visualization application with X shared memory extensions			

From Table 2, we note that with X shared memory extension support, not only the visualization application has a higher frame rate on its own, but there is also an improvement in overall performance when both applications run. That is, only 10% of frames are dropped by the video application and the frame rate increased from 26 to 35 frames per second for the visualization application. The use of shared memory significantly reduces kernel-user interactions by eliminating data movement by IPC. As a result, less time is spent in system mode (50%) thereby increasing the availability of CPU for user mode (41%). This obviously benefits the video application. Also, the frame rate increase for the visualization application can be explained by the fact that with shared memory it requires 54% of CPU in system mode as opposed to 66% when not using X shared memory.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we demonstrate that to achieve high application-application throughput in a high-speed network environment, we need to solve bottlenecks at *all* levels: network, operating system, and application. We have shown how using various optimization techniques, it is possible to increase network-application performance. These techniques include the use of a Gbit/s network, single-copy schemes (including improved protocol processing), and X shared memory extensions. Figure 8 summarizes the throughput optimizations for the visualization application. At each level of the data path between network and application, we apply the optimizations from the previous level. Thus, the final throughput of 105 Mbits/s for Jetstream is the result obtained after applying optimizations at all levels. Compare this to the results shown in Table I, where without any optimization, the throughput was 8.4 Mbits/s for Ethernet, 23.1 Mbits/s for FDDI, and 46.2 Mbits/s for Jetstream.

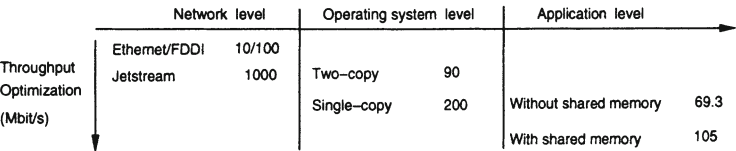


Figure 8 Summary of all applied optimizations. Each level includes the optimizations from the previous level.

As pointed out in section 4, the hardware architecture prevents the set up of a direct data path between network adapter and display for those network applications that require minimal or no data processing. Furthermore, as depicted in Figure 9, the system and memory bus controller interconnects CPU, memory, and the I/O subsystem, thereby typically becoming the bottleneck during concurrent access or transfer of data between these components. This can limit the performance of network multimedia applications which involve *simultaneous* data transfer from network to main memory, and from memory to display device.

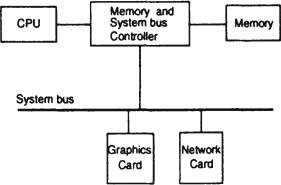


Figure 9 Simplified HP 9000 Series 700 architecture.

We are investigating new architectures which will better cope with the demands of multimedia applications in the context of high-speed networks. We anticipate that new switch-based bus architectures will allow greater flexibility in setting up different data paths between components of the system. Historically, switching logic and interconnect

components were expensive thereby limiting their use in systems. However, progress in silicon and packaging technology has changed the relative cost of interconnections and now it is possible to build general purpose computer systems based on switched bus architectures (Boxer, 1995). In this context, the data path used to derive Equation 1 no longer holds since in the case of these architectures, many paths can be used simultaneously to improve performance.

We believe that the next generation of networked multimedia applications will require more than just network displays: in addition, it should be possible to *manipulate* the multimedia data before storing, displaying or transmitting over the network. In this context, we are exploring a design space that will provide the user with the capability of setting up data paths between devices and also the flexibility of selecting portions of multimedia data in transit (e.g. from network card to display) and performing any manipulation required.

ACKNOWLEDGMENTS

The authors wish to thank the many employees of Hewlett Packard laboratories, Bristol, UK for their support and encouragement during the course of this project, in particular we are grateful to Aled Edwards for his valuable discussions on many aspects of this work. We thank Dr. Ulrich Neumann for his help in developing the visualization application. We also thank Kaleb Keithley of The X Consortium for his explanations on the X shared memory extensions. We are grateful to the anonymous reviewers for their comments on the paper. This research has been funded in part by the Integrated Media System Center, a National Science Foundation Engineering Research Center with additional support from the Annenberg Center for Communication at the University of Southern California, the California Trade and Commerce Agency, and the DARPA POLO consortium agreement MDA972-94-3-0038.

REFERENCES

- Boxer, A. (1995) Where buses cannot go. *IEEE Spectrum*, 41-45.
- Corbet, J. and Packard, K. (1991) The MIT Shared Memory Extension. *MIT Consortium*.
- Dalton, C., Watson, G., Banks, D., Calamvokis, C., Edwards, A. and Lumley, J. (1993) Afterburner. *IEEE Network*, Vol. 7 No. 4, 36-43.
- DeBaets, A. and Wheeler, K. (1992) Midrange PA-RISC Workstations with Price/Performance Leadership. *Hewlett-Packard Journal*, 6-11.
- Dittia, Z., Cox Jr., J. and Parulkar, G. (1995) Design of the APIC: A High Performance ATM Host-Network Interface Chip, in *Proceedings of IEEE INFOCOM*.
- Druschel, P. and Peterson, L. (1993) Fbufs: A High-Bandwidth Cross-Domain Transfer Facility, in *Proceedings of Fourteenth Symposium on Operating System Principles*.
- Fall, K. and Pasquale, J. (1993) Exploiting In-Kernel Data Paths to Improve I/O Throughput and CPU Availability, in *Proceedings of Usenix Winter Technical Conference*.
- Frink C., Hammond, R., Dykstal, J. and Soltis D. (1992) High-Performance Designs for the Low-Cost PA-RISC Desktop, *Hewlett-Packard Journal*, 55-63.
- Hewlett-Packard (1992) HP Visual User Environment 3.0 User's Guide, Hewlett-Packard Company.
- Jacobson V. (1990) Efficient Protocol Implementation, *ACM SIGCOMM Tutorial*.

- Jacobson, V., Braden R. and Borman D. (1992) TCP Extensions for High Performance, *RFC 1323*.
- Jones, R. (1993) Netperf: A Network Performance Benchmark, Revision 1.7, *Information Networks Division, Hewlett Packard*.
- Kanakia, H. and Cheriton, D. (1988) The VMP Network Adapter Board (NAB): High-Performance Network Communications for Multiprocessors, in *Proceedings ACM SIGCOMM, Symposium on Communication Architectures and Protocols*, 175-187.
- Keller, R., Effelsberg, W. and Lamparter, B. (1993) Performance Bottlenecks in Digital Movie Systems, in *Proceedings of the 4th International Workshop on Network and Operating System Support for Digital Audio and Video*, Lancaster House, Lancaster, UK, 163-174.
- Murphy, B. J., Zeadally, S. and Adams, C. J. (1996) An Analysis of Process and Memory Models to Support High-Speed Networking in a UNIX Environment, in *Proceedings of Usenix Winter Technical Conference*.
- Ousterhout, J. K. (1990) Why Aren't Operating Systems Getting Faster as Fast as Hardware ?, in *Proceedings of Usenix Summer Conference*, 247-256.
- Parallax Hardware Guide (1994). XVideo700, MultiVideo700, and PowerVideo700, Parallax Graphics Inc, Santa Clara, CA.
- Pasquale, J., Anderson, E. and Muller, P.K. (1994) Container Shipping - Operating System Support for I/O Intensive Applications, *IEEE Computer*, Vol. 27 No. 3, 84-93.
- Pasquale, J., Polyzos, G., Anderson, E. and Kompella, V. (1992) A Digital Video-conferencing Experiment Using DECstation 5000 Workstation and an FDDI Network, *Internal Report*, Department of Computer Science and Engineering, University of California, San Diego, CA.
- Patel, K., Smith, B. and Rowe, L. (1993) Performance of a Software MPEG Video Decoder, in *Proceedings of First ACM International Conference on Multimedia*, Los Angeles, CA, 75-82.
- Rowe, L. and Smith, B. (1993) A Continuous Media Player, in *Proceedings of the 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, Lecture Notes in Computer Science, Springer Verlag, Berlin, 376-386.
- Watson, G., Banks, D., Calamvokis, C., Dalton, C., Edwards, A. and Lumley J. (1994) AAL5 at a Gigabit for a Kilobuck, *Journal of High Speed Networks*, Vol. 3 No. 2, 127-145.

BIOGRAPHY

Sherali Zeadally is a researcher in the Electrical Engineering Department at the University of Southern California. He received the B.A. degree in Computer Science from University of Cambridge, England, in 1991, and the Ph.D. degree in Computer Science from University of Buckingham, England, in 1996. His current research interests include operating systems internals, distributed systems, high speed networks and multimedia.

Grig Gheorghiu is a Ph.D. candidate in the Computer Science Department at the University of Southern California. He received the B.S. degree in Computer Science from University of Bucharest, Romania, in 1993. His research interests include distributed systems and multimedia applications over high speed networks.

Anthony F. J. Levi is a Professor of Electrical Engineering at the University of Southern California. He received the B.S. degree from the University of Sussex, England in 1980 and the Ph.D. degree from University of Cambridge, England, in 1982. He is a Fellow of the Optical Society of America and a member of the American Physical Society. From January 1984 to mid-1993 Dr. Levi worked at AT&T Bell Laboratories, Murray Hill, New Jersey. In mid-1993 he left AT&T to take up a position as Professor of Electrical Engineering at the University of Southern California. Professor Levi's research interests include scaling of photonic devices to sub-micron dimensions, integration of electronic and photonic devices for high-speed network applications, and optimization of networks for high-sustained throughput applications. To date, he has published over 150 refereed journal papers and holds 9 U. S. patents in these and related research subjects.

Issues in Platform-Independent Support for Multimedia Desktop Conferencing and Application Sharing

O. Kim, P. Kabore, J.P. Favreau and H. Abdel-Wahab
Multimedia & Digital Video Technologies Group
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, Md 20899, USA
(okim,kabore,favreau,wahab)@snad.ncsl.nist.gov

Abstract

Although Multimedia desktop conferencing and application sharing among geographically dispersed users are increasingly popular modalities, their spread is inhibited by platform-dependency problems. In this paper, an approach which exploits the use of the Java programming language to accommodate different hardware and window systems is investigated and a prototype is implemented. Our approach is based on *replicated tool architecture* in which each participant runs a copy of the application and the activity of each user is multicast to all the participants in the conference. The problems associated with this approach such as view synchronization and replicated object management are among the issues addressed in our research. In addition, we are developing standard functions and mechanisms that allow conference participants to seamlessly use the audio and video features available on most PC's and workstations. Our research on multimedia stream synchronization and adaptation, the incorporation of reliable multicasting and the development of distributed control algorithms are expected to result in increased conference quality, performance and robustness.

Keywords

High Bandwidth Applications, Multimedia Communications, Java, Interoperability, Computer Supported Cooperative Work, Desktop Conferencing, Multicasting, Distributed Systems.

1 INTRODUCTION

With the proliferation of high bandwidth computer networks and powerful multimedia workstations, it is now feasible to build collaborative systems that allow users to have real-time interaction with each other and remotely work together as a team. In addition to using audio and video we believe that it is very productive if all participants simultaneously have full access to their shared computer-stored materials and have the ability to share and manipulate them together.

Most current existing collaborative systems require the participants in a conference to use the same window system. For example, XTV (Abdel-Wahab *et al.* 1991, Abdel-Wahab *et al.* 1994) and Suite (Dewan *et al.* 1993) are based on the X window system and require that the participant's machines run the X server. Other systems such as WTV (Adams 1995) have tried to replicate the functionality of XTV replacing the X windows with Microsoft Windows. Ideally, each participant in a collaborative conference should be able to use whatever platform he or she prefers. For example, some may use PCs running MS Windows 95. Others may use workstations running different version of UNIX and X windows, yet others may use PowerPC Macintoshes. Before the introduction of Java, this sort of collaboration was enormously difficult to achieve. Java programs are compiled to an architecture neutral byte-code format and thus can run on any system that implements a Java virtual machine and its abstract window system. Java provides a fortuitous opportunity for the Computer Supported Cooperative Work (CSCW) (Grudin 1994) community to overcome a barrier which hitherto hindered the wide spread use of collaboration technology.

To overcome the platform-dependency problem for application sharing in heterogeneous platforms, NIST (National Institute of Standards and Technology) and ODU (Old Dominion University) are jointly conducting a research project to investigate mechanisms for sharing multimedia applications among participants on not only heterogeneous windowing and operating systems, but on different hardware platforms.

We have developed mechanisms to intercept, distribute and recreate the user events that allow single-user Java applications to be shared, without modifications, among conference participants. These mechanisms can be run transparently on any system implementing Java. The mechanisms incorporate the services of network communications, conference management and floor control management. The network communications services include distribution of the data among conference participants; conference management includes joining and leaving a session; and floor control includes participant's control and interaction with the application during a session.

In this paper, we refer to the prototype which has been developed as the Java Collaborative Environment (JCE). We are now in the process of aug-

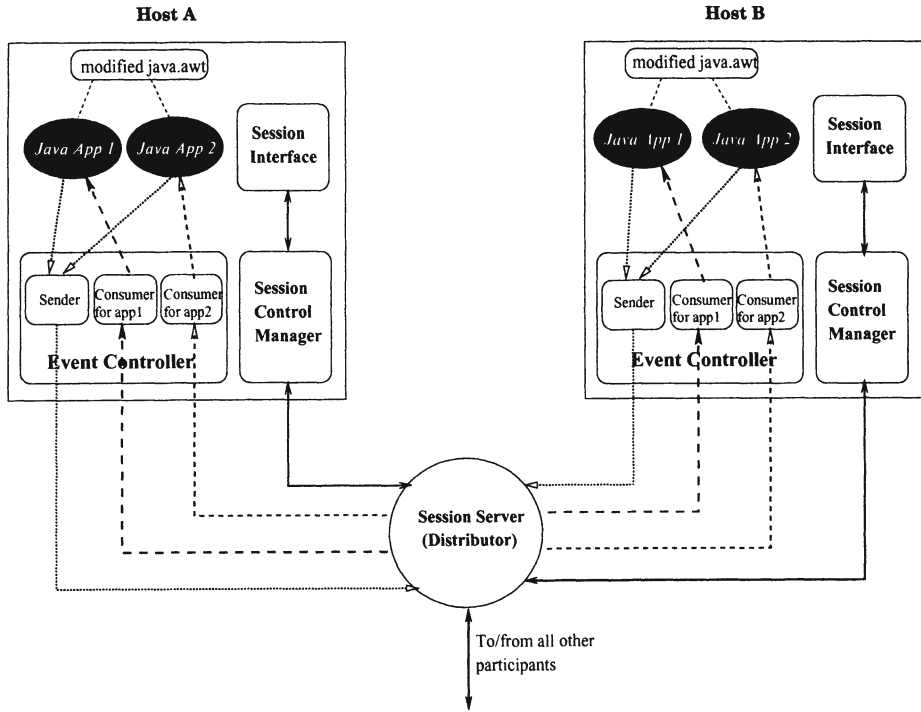


Figure 1 Overall System Architecture.

menting JCE to include both audio and video in an integrated platform-independent desktop conferencing system.

The remainder of the paper is organized as follows: Section 2 describes the JCE system architecture and provides an overview of its major components and functions. Section 3 discusses the problems associated with executing multiple replicated copies of the same programs. In Section 4 we show some applications for event logging: late-joining and playback. Section 5 concentrates on communications issues such as the use of reliable multicasting, conference information service and conference control. Support for platform-independent audio and video is discussed in Section 6. The quality of session attributes such as multi-media synchronization and adaptation are presented in Section 7. Finally, Section 8 gives our conclusions and future work.

2 JCE SYSTEM ARCHITECTURE

The JCE is a framework for shared interactive multimedia applications. Figure 1 depicts the overall system architecture, and the relationship and communication paths among all processes of the system, for a given conferencing session. The Java applications denoted as *Java App 1* (and 2) in Figure 1 are not part of the system. They are collaboration-unaware single-user applications

developed using the standard `java.awt` package (AWT 1995). Participants can invoke one or more applications in a given conference. Our model is based on the *replicated architecture* (Steinmetz *et al.* 1995) in which an instance of each application runs locally at each participant's site and the activity of each user is distributed to all the participants in a conference.

As shown in Figure 1 the JCE provides a *modified java.awt* class library for use at runtime and consists of three components: the *Session Control Manager* and its Interface, the *Event Controller*, and the *Session Server*. These components are discussed in the next two subsections.

2.1 Modified Java Windowing Package

In the standard `java.awt` package (AWT 1995), **Component** class is the superclass of all the GUI components and it contains the user event handling. The unmodified *handleEvent* method in **Component** class processes the user events. We have modified this *handleEvent* method to provide a mechanism that intercepts the user events from each application, and sends them to the Session Server, giving all participants the same application state.

The following code fragments show the modified *handleEvent* method in **Component** class, with modifications shown in *italic*.

```
import EventController;

public class Component implements ImageObserver {
    // existing code
    public boolean handleEvent(Event evt) {

        if (!EventController.sender(evt))
            return false;

        switch(evt.id) {
            // remaining existing code
        }
        return false;
    }
}
```

This approach allows Java single-user applications to be shared, and enables their simultaneous viewing and collaboration among conference participants.

2.2 System Components

The *Session Control Manager* (SCM) and the *Session Interface* combined provide the user with a graphical interface offering the following options: to call, join or leave a session; to start applications; and to request or release a floor. Each participant is given an SCM process that exchanges control information with the Server for the duration of the session.

The *Event Controller* is the core of the collaboration mechanisms. It is composed of two processes: the event *Sender* and *Consumer*. When an application is started an *Event Controller* for the application is automatically instantiated by the Session Control Manager. When two or more applications are shared, two or more *Consumers* are created as shown in Figure 1. The *Sender* is declared as a static (i.e., class) method of the *Event controller*, so only one *Sender* method exists for all applications. The *Sender* method first checks the intercepted event to determine whether or not it should be sent to the *Session Server*, since events originating from shared applications are always forwarded. The *Consumer* processes receive events redistributed by the Session Server from other participants, and post them to the local instance of the application as if they were originated locally. This process is completely transparent to the application, i.e., the application is unaware that it is being shared.

The *Session Server* in Figure 1 provides three distinct functions: distribution of all messages to all participants; group management for a given session, including joining or leaving a session; and server floor control management.

2.3 Alternative Implementations

Besides the modifications to the standard java.awt package (AWT 1995) which allow existing single-user applications to be shared, as detailed above, we have also developed an alternative collaboration mechanism to intercept the user events, which is the extensions to the standard java package. The extensions called *collawt* allow application developers to develop new collaboration applications or modify existing single-user applications (Abdel-Wahab *et al.* 1996a). The advantages of each approach are noted below.

Advantages of Modified Library

The existing and new single-user applications can be shared transparently, so that application developers do not have to be concerned about whether the applications are collaboration-aware whereas under *extended* approach, new collaborative applications importing *collawt* need to be developed, or existing applications modified, if this is possible, to enable collaboration.

Further, since **Component** class is the superclass of all the GUI components, the JCE need not be updated when new GUI components are developed and introduced. In contrast, the *extended* libraries must be updated to account for the new components.

Advantages of *Extended* Library

This method provides more efficiency and flexibility in object event handling in shared applications. Each GUI component in *collawt*, derived from the **Component** superclass, handles its own user events, thus eliminating those events coming from other than shared applications such as the *Session Control Manager*.

Moreover, the *extended* library requires that no change needs to be made to the environment by changing the java **CLASSPATH** variable, whereas the use of the *modified* library requires that the new java.awt package must be installed and used at runtime.

3 REPLICATION MANAGEMENT

Most applications need to create or use objects during execution, for instance, the environment variables, the initialization dot files, and the files storing multimedia data. These objects must be replicated and available at each site before the invocation of an application for the correct operation of the JCE system. There are three types of objects to be replicated and managed: environment, operational and final objects.

3.1 Environment Objects

In order to enforce WYSIWIS (What You See Is What I See), each replica of the shared application must have the same *operating environment* (e.g., in UNIX/X systems terminology, each site should have the same environment variables, same initialization files and the same X resource files). Before the invocation of each copy of the shared application at each site, we must ensure that all sites have identical operating environments. Since each application may have a specific and different operating environment from other applications, any solution to this problem requires obtaining specific information about each application. This can be achieved by having an *environment profile* for each shared application that contains the operating environment specification and default locations where the values of these resources can be obtained, replicated and installed at each participant's site prior to the execution of the application. It is prudent to save and later restore the original

operating environment so that the user can run the application in single-user mode according to his/her own preferred settings of the application. This concept of common operating environment and profile for each shared application is important to guarantee full WYSIWIS behavior. This does not preclude the participants from sharing applications in which users see the same data in different ways such as local selection of font styles and sizes according to each user's preference.

3.2 Operational Objects

The second set of objects needed during the life-time of shared applications is the *operational files*. These files may include data, images, audio clips, video clips, etc. that are needed during the execution of the program. If these files are known and available in advance, then we can specify an *operational profile* for each application that contains a list of these files and a default location where it can be obtained. Prior to the execution of the application, these files are distributed and installed in the appropriate "well-known" directories. Again, one ought not to destroy or alter the original copies of these files so they can be restored upon the termination of the shared application.

3.3 Final Objects

The third set of objects is the newly created files or the files to be modified during the shared application life-span. In this case a *final profile* is used to list these files and specify whether each participant should keep a copy. It may be necessary (e.g., for integrity or security reasons) to specify for some files that only one site should keep a copy and that all other copies of these files should be deleted.

4 LATE COMERS, RECORDING AND PLAYBACK

Saving the user input events to each shared application by the conference server is called *event logging* and this is similar to transaction logging in database systems. There are many applications for event logging in JCE:

1. *Late Comers*: Participants who join an ongoing conference after the start of at least one shared application are called *Late-Comers* (Chung *et al.* 1993). Although in JCE, participants may join the conference any time after it starts, they may not have the same view for those shared applications which started before they joined. To bring the late-comers views in synchronization with all other participants, we may send all the logged events to their

instances of those shared applications. However, saving all the user events from all participants can be very inefficient, since a significant amount of memory space and network bandwidth are required. Therefore, only the user action events such as keyboard events and drawing lines meaningful to each shared application are stored and indexed. The mouse motion events such as `MOUSE_MOVE` and `MOUSE_ENTER` are not logged.

2. *Recording/Playback*: Event logging is considered to be a form of session recording. To playback a recorded session, all it takes is to start an instance of each involved application and feed it with the events saved in the log file. The playback may be seen by a single person or by all the participants in a conference like any other “live” shared application. This can be very useful in many applications such as:

- To investigate why an application has crashed and the sequence of events that has led to it.
- To use it as a teaching aid by recording the steps of interaction with an application which users may view at a later time.

5 COMMUNICATION AND DISTRIBUTED CONTROL ISSUES

This section is devoted to issues of system performance, usability and robustness. To increase the system performance as the number of participants increases we should use reliable multicasting for data transport instead of the current server-based (star-topology) TCP connections. To make it easy for participants to use the system and join any on-going conferences or start new conferences, we should use the services provided by CIS, a real-time, internet-based Conferencing Information Server, as discussed in Section 5.2. A robust conferencing system should not depend on one central process for its control so a set of distributed algorithms must be developed to replace the functions performed by the current conference server.

5.1 Use of Reliable Multicasting

In our current implementation, all the participants are connected to the server with TCP connections (Postel 1981) as shown in Figure 2. If one participant needs to send a message to all other participants, he or she sends it to the server which in turn distributes it to all participants, one at a time, using the TCP connections. This may be acceptable if the number of participants is small (e.g., 4 or 5). However, as the number of participants increases, the system performance degrades and the session quality is reduced, as measured by several parameters, such as view synchronization, to be discussed in Section 7.1.

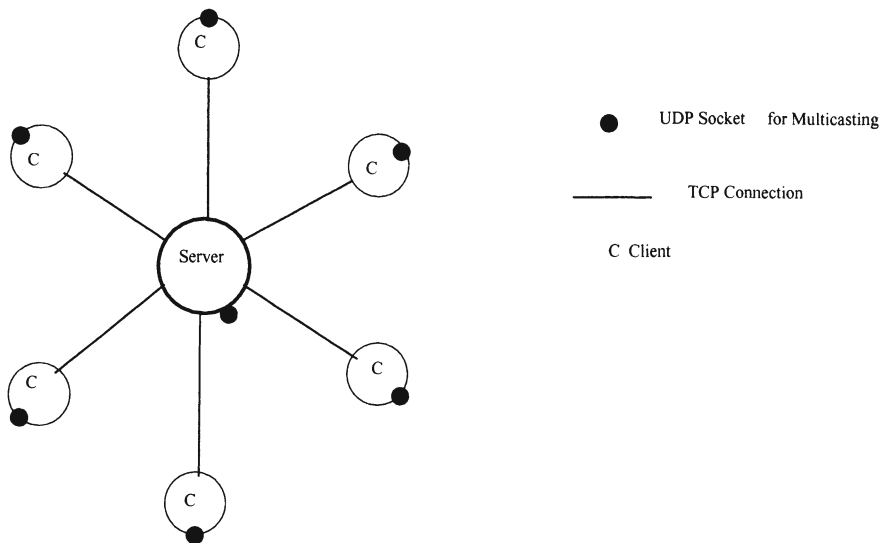


Figure 2 Communications among Clients and Server.

The use of *reliable multicasting* (e.g., RMP provided by Berkeley/West Virginia (Whetten *et al.* 1994)) greatly improves both the performance and the quality of session. To use multicasting, each participant and the server will have a UDP socket in addition to the existing TCP sockets as shown in Figure 2. TCP connections are used for one to one communications among the participants and the server. In the current JCE implementation if the server is down the whole conference will be terminated and can not continue. However, as discussed in Section 5.3, we plan to distribute the server functions among all the participants and eliminate the need for TCP connections. At that time we will have a truly distributed conferencing system that is not subject to a single point of failure.

5.2 The Conferencing Information Service

In order to facilitate the process of joining a conference, a conferencing information service (CIS) (Abdel-Wahab *et al.* 1996c) is utilized. This service allows a conference to advertise specific information about itself to help potential participants find out information about the conference and allow them to join. In order to use CIS, a conferencing system like JCE needs to implement the CIS advertisement protocol and provide an interface that allows users to browse through the information about various conferences and join any selected conference. This interface may be implemented as a stand-alone application or as a Java applet that can be used within an Internet browser.

5.3 Distributed Management of Conference Resources

In our current architecture the server plays a central and vital role in connecting the participants, using the star-topology TCP connections shown in Figure 2, and performing various conference management functions such as floor management. As we have seen earlier, we can use reliable multicasting to replace the role of the server to distribute messages among the participants. We have developed a couple of distributed algorithms that are needed to:

1. maintain an up-to-date list of conference participants and to announce this list to the Conferencing Information Server; and
2. to grant the floor to at most one participant at a time.

6 PLATFORM-INDEPENDENT AUDIO AND VIDEO SUPPORT

Beside shared applications, audio followed by video in this order are important to support full and effective collaboration among participants. In our project, we provide audio support as a standard feature, since the Internet bandwidth may now reasonably support the transport of audio conversations.

Almost all PCs and Workstations now have audio devices (microphone and speakers), though they are often not compatible with each other and may use different audio formats. Thus, our task here is to ensure that all participants can talk and hear each other without worrying about the heterogeneity of their respective audio devices. Our approach to resolving this issue is to identify the most common audio format and configuration of audio devices and save it as a *Common Audio Format and Configuration* (CAFC) file for each operating platform. Whenever a participant joins a session, his/her audio devices are examined to see if they can be configured to the specifications stored in the CAFC file for the specific operating environment. If it is determined that a participant's device cannot be configured or does not support the common audio format specified in the CAFC file, then an appropriate action such as format translation or quality of service degradation for that participant is taken.

Despite advances in compression technology, video communication requires high bandwidth and not many PCs or workstations are equipped with video cards and cameras, which are still expensive components relative to the basic price of the host machines and must be purchased and installed separately. However, we expect in the near future that Internet bandwidth will increase and the video hardware cost will decrease to the point where desktop video communication will become as common as audio. In a two-party system, it is customary to display the other person's video image. When there are multiple participants, however, determining which participant's video image is to be displayed at which time is a matter to be decided by each specific application. For example, in the Interactive Remote Instruction (IRI) system

(Abdel-Wahab *et al.* 1996b) used for distance learning, the teacher's image is always displayed on each student's workstation and only those students engaged in active discussion with the teacher are displayed, in smaller windows.

In a general desktop conferencing system, we would like to provide general mechanisms and protocols that users can configure according to their particular needs and preferences. For example, if someone speaks, his image may be displayed by clicking a button if so desired. In a formal meeting where there is a chairman, the group may decide that the chairman's image be always displayed. This issue of determining how many images to display, the quality and size of each image, and when these images are to be displayed is one of the goals of our project. Our other major goal is to support interoperation among many different and diverse video devices by providing Java programs as interface for multiple cards using different hardware platforms. If some participants have no, or incompatible video capabilities, they can still participate in the conference using only the audio channels.

To achieve maximum efficiency, in our implementation, we intend to use the traditional IP multicasting to send audio data among the participants (Casner *et al.* 1992). In addition, the standard IGMP (Internet Group Management Protocol) (Stevens 1994, Deering 1989) will be used to manage the process of joining and leaving a conference.

7 SYNCHRONIZATION AND ADAPTATION ISSUES

In order to improve the conference quality as perceived by the participants we must address the following issues and search for innovative solutions.

7.1 View Synchronization

As we described earlier, the JCE is based on the replicated model (Steinmetz *et al.* 1995); that is - for n participants there are n copies of the same application running concurrently, possibly on different hardware (e.g., from powerful workstations to low-end PCs) and software (e.g., operating systems and window systems). Some of the n participants may be connected by a high-speed Intranet, while others are connected to the global Internet with relatively slow links. In this networking environment, it is inevitable that there will be a skew or lack of synchronization among what all the participants see in the shared application windows. Our objective is to reduce these synchronization problems to a minimum and bring it to an acceptable human tolerance level to preserve the concept of WYSIWIS. This problem does not exist in a two-party conferencing system. However, when a large number of participants is involved, the problem is significant and requires an innovative solution. One such solution is to sense and measure the state of each replica of the shared application. The gathered feedback data can then be used to slow down the

flow of events to the faster participants or to speed up the delivery of events to the slower sites.

7.2 Multi-Stream Synchronization

An important problem in multimedia applications (e.g., remote learning, video conferencing and information-on-demand) is the temporal synchronization of continuous and discrete media that have the same or different sources. Streams can be captured at the transmitter and a temporal relation between them established. The playback times, at the destination, for the corresponding streams may differ due to communication and network delays, or the difference between two consecutive schedule times of a process, for example. Temporal synchronization requires the preservation of the temporal dependencies among various media at the destination. For example, consider the following scenario. In a collaborative session, at time t_0 , a participant speaks, then, at a later time t_1 , he/she starts a shared Java application (e.g., white-board). On other participant's workstations, it is not sufficient just to play the streams, temporal synchronization must also be maintained. Within $[t_0, t_1]$, the speaker's audio and video need to be synchronized, and the time independent stream generated by the Java application should be synchronized with the other two continuous streams.

There are two particular issues that need to be addressed for temporal synchronization: *intrastream* synchronization and *interstream* synchronization (Steinmetz *et al.* 1995). Intrastream synchronization policies eliminate jitter when playing a periodic stream. Interstream synchronization policies support orchestrated multimedia presentations, preserving the time dependencies between streams when captured. The relations that specify the temporal dependencies between streams are called *synchronization specification* (Steinmetz *et al.* 1995). In live synchronization, the application at the transmitter is responsible for providing the synchronization specification, while the application at the destination is responsible for providing a synchronized presentation according to this information.

The objective of our work is to provide a flexible and robust solution for the temporal interstream synchronization of time dependent (audio, video) and time independent (text, graphics, shared windows) streams in a multimedia application.

7.3 Inter-Stream Adaptation

In collaborative multimedia systems, there is a need for overall control, beyond the level of quality of service (QoS) of individual streams. The quality of the conference as perceived by the end user, must be determined by the end

application. At every instant in time, the quality of the conference depends on the priorities of the on-going streams, from the user's perspective, as well as on the actual QoS offered by the system to each of these streams. The main objective is to keep a collaborative session going, with acceptable overall quality. This is achieved by employing a monitoring mechanism at the application level for monitoring the perceived QoS of each stream. For example, a two way audio-video application may choose to degrade the quality of video only, while keeping the audio quality at the same high level.

A system which is not aware of this inter-stream correlation, may degrade the performance of all streams with an equal proportion in an attempt to react to the overload situation in a fair way. Moreover, the same application may have different priorities for different streams, at every instant in time. Building on the same example mentioned above, if there were a conversation between two physicians, and at a certain point in the conference, the video image of one of the participants was replaced by a VCR tape playback of an operation, then the application may prefer a degradation in the quality of the audio rather than that of the video in reaction to any overload situations. In such complex collaborative applications, a compromise in the quality of one stream in favor of another may not only be due to temporary overload situations, but also to inherent capacity constraints in the system. For instance, a video conferencing application supporting several simultaneous participants, may not find enough network bandwidth, or system processing capability, to send a full motion video stream of each participant at 30 frames per second. As an alternative, each participant may receive a full motion video stream for the speaker, and a lower frame rate video streams for other participants. The previous examples suggest that, in collaborative multimedia systems, there is a need for overall control, beyond the level of QoS of individual streams, for a particular application.

8 CONCLUSIONS AND FUTURE WORK

In this paper, we have described our current ongoing research and the major issues and problems associated with developing platform-independent desktop conferencing systems that integrate application sharing, audio, video and conference management functions. Among those issues addressed in some detail are replication management, accommodating late comers, session recording and playback, scalability through the use of reliable multicasting in both reliable and unreliable (raw IP) forms, global internet conference information service, the integration of audio and video, and the synchronization and adaptation of multimedia streams.

In addition, we have demonstrated the important role of Java by implementing the Java Collaborative Environment (JCE) prototype for application sharing among diverse systems such as UNIX workstation-based and PC Windows-based systems. The merits of the two alternative collaborative mech-

anisms developed in JCE, the *modified* and the *extended* approaches are also discussed. Our next goal is to use JCE from Internet browsers such as the Netscape Navigator and the Microsoft Internet Explorer. Due to some of the limitation imposed by Internet browsers and Java Applets for security and other reasons, the participants may not be able to perform certain functions. However, we would like to maximize what can be done through the World-Wide Web, identify those functions that cannot be performed through it and help the users to perform these functions via a parallel stand-alone interface.

REFERENCES

- Abdel-Wahab, H. and Feit, M. (1991) XTV: A Framework for Sharing X Window Clients in Remote Synchronous Collaboration, *Proceedings, IEEE TriComm '91: Communications for Distributed Applications & Systems*, Chapel Hill, North Carolina, pp. 159-167, April 1991.
- Abdel-Wahab, H and Jeffay, K. (1994) Issues, Problems and Solutions in Sharing X Clients on Multiple Displays, *Journal of Internetworking Research & Experience*. pp. 1-15, Vol. 5, No. 1, March 1994.
- Abdel-Wahab, H., Kvande, B., Nanjangud, S., Kim, O., and Favreau, J.P. (1996a) Using Java for Multimedia Collaborative Applications, *Proceedings, PROMS'96: Third International Workshop On Protocols for Multimedia Systems*, 1996.
- Abdel-Wahab, H., Maly, K., Youssef, A., Stoica, E., Overstreet, C.M., Wild, C., and Gupta, A. (1996b) The Software Architecture and Interprocess Communications of IRI: an Internet-based Interactive Distance Learning System, *WETICE'96*, Stanford, California, June 1996.
- Abdel-Wahab, H., Stoica, I., and Sultan, F. (1996c) The Design and Implementation of an Internet Conference Information System, To appear in *Journal of Internetworking Research & Experience*, 1996.
- Abstract Windowing Toolkit (AWT) package, Java Developers Kit (JDK) Version 1.0 API*, (1995) Sun Microsystems Inc. Mountain View, CA 94043
- Adams, D. (1995) WTV: An MS Windows based Collaborative System, Master's Project Report, Department of Computer Science, Old Dominion University, Dec. 1995.
- Casner, S., and Deering, S.E. (1992) First IETF Internet Audiocast, *Computer Communication Review* vol. 22, no. 3, (July), 1992
- Chung, G., Jeffay, K., and Abdel-Wahab, H. (1993) Accommodating Latecomers in Shared Window Systems, *IEEE Computers*, pp. 72-74, Vol. 26, No. 1, January 1993.
- Deering, S. (1989) *Host Extensions for IP Multicasting*, IETF RFC 1112, 1989.
- Dewan, P. and Chouldhary, R. (1993) A high-level and flexible framework for implementing multiuser interfaces, *ACM Transaction on Information Systems*, Vol. 10, No. 4, 345-380, (October 1993).
- Grudin, J. (1994) Computer-Supported Cooperative Work: History and Focus,

- IEEE Computer*, Vol. 27, No. 5, 19-26, (May 1994).
- Postel, J. (1981) Transmission Control Protocol, *IETF RFC 793*, 1981.
- Steinmetz, R. and Nahrstedt, K. (1995) *Multimedia: Computing, Communications & Applications* Prentice-Hall, 1995.
- Stevens, W.R. (1994) *TCP/IP Illustrated*, Volume 1, Addison-Wesley, 1994.
- Whetten, B., Montgomery, T. and Kaplan, S. (1994) A High Performance Totally Ordered Multicast Protocol, *Theory and Practice in Distributed Systems*, Springer Verlag LCNS 938, 1994.

9 BIOGRAPHY

Ms. Okhee Kim is a computer scientist at the Information Technology Laboratory (ITL) of the National Institute of Standards and Technology (NIST). She has been working at NIST since 1985, conducting research and development activities on multimedia collaboration, EDI tool development, GOSIP testing program, and the Transport performance measurement. She received her Masters Degree in Computer Science from New York Institute of Technology in New York, 1985.

Paul Kabore is a guest researcher at the ITL of NIST since June 1996. Before joining NIST, he was an assistant professor at the University Henri Poincare of Nancy, France and a research assistant at the CRIN/INRIA laboratory of Nancy. He received his M.S. in computer science from Institut Africain d'Informatique of Libreville, Gabon in 1991 and a Ph.D in computer science from the University Henri Poincare of Nancy, France in 1995.

Jean-Philippe Favreau is the Group Manager of the Multimedia and Digital Video Technologies Group at the ITL of NIST. He is currently leading the Secretariat for Federal EDI. Dr. Favreau has been working at NIST since January 1985, managing projects and conducting research and development activities on multimedia technologies. He received his Ph.D. in Computer Science from the University of Bordeaux, France, in June 1986 and his Master in Business Administration from the University of Maryland, College Park, in December 1991.

Hussein Abdel-Wahab received the Ph.D in 1976 and the MS in 1973 both from the University of Waterloo in Computer Communications and the BS in Electrical Engineering from Cairo University in 1969. Currently he is a full professor of computer science at Old Dominion University. In addition he is an adjunct professor of computer science at the University of North Carolina at Chapel Hill and a faculty at ITL of NIST. His current research interests are in the areas of networked multimedia and collaborative systems with emphasis on distance learning applications.

Design and implementation of a flexible traffic controller for ATM connections¹

L. LAMTI, H. AFIFI, M. HAMDI

*Ecole Nationale Supérieure des Télécommunications de Bretagne,
RSM Department*

2, Rue de la Châtaigneraie, BP 78, 35512 Cesson Sévigné, France

Phone: +33 02 99 12 70 46 / Fax : 33 02 99 12 70 30

E-mail: {Leila.Lamti, Hossam.Afifi, Maher.Hamdi}@enst-bretagne.fr

Abstract

In ATM networks, the congestion control paradigm is built on two basic principles. The first states that the adequacy of a service delivered to a particular user should not depend on the detailed behaviour of other users. The second is that network should provide enough feedback to users to effectively utilize the available resources. Traffic management is concerned with controlling the source behaviours to ensure they get their desired Quality of Service (QoS). It includes the Connection Acceptance Control (CAC) and the Usage Parameter Control (UPC). In order to avoid data loss at the UPC, we propose to implement a traffic controller at the source edge. With its two components, - a traffic regulator and a connection scheduler-, this controller shapes the input traffic so that it becomes amenable to the policing mechanisms. The purpose of locally controlling a source is the improvement of the TCP performance over ATM networks. We argue that the policing mechanism used in the regulation component should be less stringent than the Generic Cell Rate Algorithm (GCRA) proposed by the ITU. The control method proposed in this paper is more flexible and is suitable for providing performance guarantees in high speed networks. Moreover, a quantitative means of defining a packet-to-cell parameters conversion is discussed. Its purpose is to ensure the efficiency of the control mechanism when implemented at the packet level. We present the results obtained by the implementation of this controller in a multi-tasks environment. We show how we can obtain reasonable bandwidth utilization while remaining conform to the traffic contract limitations. Finally, we present how this method achieves simplicity of implementation as well as flexibility in the allocation of bandwidth to different connections.

Keywords

ATM traffic contract, Traffic shaping, Leaky Bucket algorithm, Packet-to-cell parameter conversion.

¹ This work was partially funded by a CNET-Lannion, France Telecom Grant No:94PE7301.

1 INTRODUCTION

High speed ATM networks in both local and wide environments start supporting per-connection end-to-end performance guarantees expressed in terms of delay, throughput and loss rate bounds (Zhang and Ferrari, 1994). Delivering this Quality of Service (QoS) to users in computer applications such as desktop video services raises however a certain number of problems that still have to be considered.

These problems are scattered in different protocol layers ranging from the application level down to the hardware interface through the conventional stack of protocols. They are also caused by the operating system and the interaction of task scheduling with flow control.

Problems found in the protocol stack are of different kinds. In the application layer, we face ergonomic choices like the necessary bandwidth needed for a Web session. Resource sharing among users is also an important issue in a guaranteed service network. These problems are beyond the scope of this paper. Related work can be found in (Garrett, 1996).

Interaction of the transport layer flow control and the ATM one is still an open issue. Several contributions have for example proved that TCP protocol flow control is incompatible with ATM rate based flow controls like CBR or VBR. Proposals for new transport protocols and adjustments to existing ones can be found in (Elloumi, et al., 1995). Although network layer does not directly affect traffic management through specific flow control algorithms, we find in some situations a contradiction between the need to offer a guaranteed QoS and the inherent role of multiplexing usually found in this layer. A typical example can be encountered in the IP over ATM solution where TCP and UDP traffic from different applications is multiplexed on virtual channels according to their destination. In such a solution, many problems of performance guarantee can be encountered. In fact, at the IP level, packets from different connections will interact with each other; without proper control, these interactions may adversely affect the QoS guarantees desired by each source.

On the other hand, congestion control mechanisms implemented within ATM networks rely on admission control, resource reservation and scheduling schemes (Radhakrishnan and Raghavan, 1994). Resource reservation schemes manage the allocation of the resources at each of the nodes so that per-node QoS requirements can be met for each connection. With only these control mechanisms, QoS requirements can't be satisfied. This is due to the fact that users may attempt to exceed QoS limits specified at the connection establishment within the traffic contract. In fact, it is to be noted that users (e.g. workstations) have a physical access offering a bandwidth much larger than what they may choose in terms of traffic contract parameters. So, it is an important task to control at the user side that no violations are committed before data is sent to the network. The need of traffic enforcement and shaping at the edges of the network becomes consequently important. We propose in this paper a solution to implement traffic shaping at the user's side in order to check that their flow's characteristics remain conform to the traffic contract.

We also present the principle of a packet-to-cell parameter conversion used in our implementation in order to take into account the fluctuations resulting from the fragmentation of a TCP packet into ATM cells.

The rest of this paper is organized as follows:

In Section 2, we delineate the TCP over ATM environment and its requirements. We show in this section what must be changed in TCP in order to enhance the application performances in terms of traffic enforcement. A solution consisting in a controller offering both traffic shaping and connection scheduling is then introduced.

In Section 3, we first describe the controller's architecture. We then present the Generic Cell Rate Algorithm scheme and show its negative impact on a source traffic if implemented in a multi-tasks environment. We discuss the need of a less stringent policing mechanism which allows short term burstiness and show how it can result in a considerable performance improvement. We present a solution using a credit-based scheme to implement the traffic shaping. A quantitative means of defining a packet-to-cell parameter conversion is discussed. Its purpose is to ensure the efficiency of the control mechanism when implemented at the packet level. The principles of such a conversion and its effect on the credit calculation are described.

We present in Section 4 the controller's implementation and discuss its ability to smooth the burstiness of a source input traffic while respecting the bounds of the traffic contract. A comparison is made in this section between the performances obtained with the GCRA and with the proposed controller. Finally, Section 5 summarizes and concludes this paper.

2 THE TCP OVER ATM ENVIRONMENT

2.1 Requirements:

Referring to the IP over ATM working group recommendations (IP-over-ATM WG, 1995), "ATM networks should provide a class of services which strives to cooperate with existing TCP congestion avoidance mechanisms, thereby, explicitly providing support not only for directly-attached and aware endstations, but also for endstations on LANs that are using current TCP implementations".

It is becoming more common that large ATM networks provide virtual channels for the IP layer services in addition to native ATM services. In fact, the ATM alternative as a new support for computer communications has demonstrated some trials for the implementation of new transport protocols over native ATM (Cole, et al., 1995). The main motivation behind these trials is that TCP/IP fails to exploit ATM benefits (high bandwidth, QoS guarantees..). Some of these proposals consider also the use of a dual stack (either TCP/IP or native ATM transport protocol) (Almesberger, 1996) that starts a connection using conventional TCP/IP and switches if possible to an ATM native connection.

This paper presents a different approach that keeps the current TCP/IP protocol stack. We introduce however some necessary modifications to deal with native ATM connections. The advantages of this solution and additional protocol design details can be found in (Afifi, et al., 1996).

2.2 The self-regulation mechanism

In ATM networks, a call setup is always associated with a set of parameters specifying a service rate (even if no service rate is required by the application). If the call is accepted, the network guarantees the bandwidth. It also verifies the non-violation, by the user, of the connection parameters. This constitutes a traffic contract used to apply a control mechanism called *Usage Parameter Control* (UPC). ATM cells for a given connection are checked. Conforming cells are routed towards their destination. Others are usually discarded.

TCP protocol implements a specific congestion control mechanism designed for packet networks with no bandwidth or delay guarantees (Jacobson, 1988). This algorithm has been progressively improved (Matthis, et al., 1996) and is perfectly designed for communications needing a tight closed loop flow control. TCP/IP can be used for ATM networks with the help of the "Classical IP over ATM" framework (Laubach, 1994). In this case, it is recommended to use a best-effort connection. It is possible to offer a global guaranteed rate to all of the TCP/IP traffic (with Classical IP over ATM) going from a given source to a destination by pre-establishing the necessary virtual circuit with QoS guarantees. It is however not possible to establish separate TCP/IP connections each with a different guaranteed traffic contract. The reason comes from 1) TCP/IP flow control that acts separately for each connection. Connection starting late (a situation where start time arrives when other connections are already in an advanced Slow-Start region) or subject to losses will negatively affect the corresponding throughput 2) The global pre-established virtual circuit controlling the multiplexed IP traffic that is not capable of dealing with separate connections.

We propose to adapt the TCP/IP environment to deal with ATM requirements. By forcing self-regulation for each source, we avoid the loss of non-conforming packets through the ATM network and the unfairness resulting from the separate, independent closed loop flow control.

3 THE CONTROLLER'S ARCHITECTURE

As shown in (Figure 1), we have divided the traffic controller into two components: a traffic regulator and a source scheduler. The purpose of such a decomposition is to offer an independent regulation for each connection according to its traffic parameters. Thus, we immunize the bandwidth allocation from the effect of end-to-end delay variation among all the sources. The regulator shapes the input traffic on each connection into the desired traffic pattern. The regulation is performed by a flexible traffic shaper using a credit-based scheme. The regulators achieve the control before handing the input traffic to the Scheduler. The Scheduler is composed of real-time and non-real-time queues (Operating system Streams deal with pointers only, no copies are made within the kernel). A connection is considered to be real-time if it negotiates a CBR or VBR traffic contract.

The UBR connections are considered non-real-time. The most common First Come First Served (FCFS) discipline will be used for both queues. The role of the Scheduler appears only when the host's load prevents from fulfilling all the required QoS guarantees.

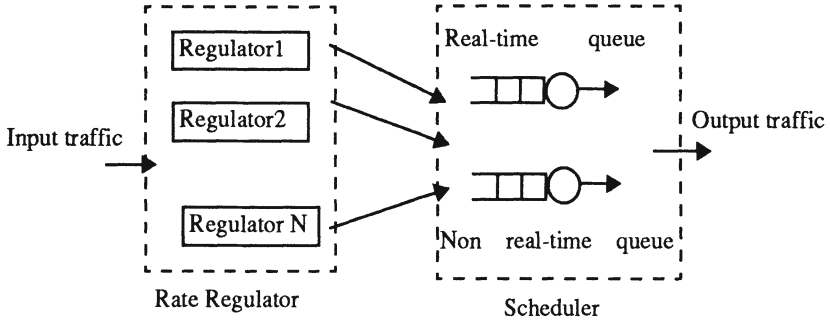


Figure 1 The controller's architecture.

3.1 The regulator's design

The GCRA scheme

Data flow generated by a source is examined within the UNI/NNI according to the Generic Cell Rate Algorithm (GCRA) (Sathaye, 1996). An equivalent traffic shaping must be performed at the source edge. We first present the principle of this algorithm and then explain how it can be considered too stringent if applied as a traffic shaping scheme.

The GCRA is defined with 2 parameters: the Increment (I) and the Limit (L). It can be a virtual scheduling algorithm or a continuous-state leaky bucket algorithm (Sathaye, 1996). The virtual scheduling algorithm updates a Theoretical Arrival Time (TAT), which is the "nominal" arrival time of the packet assuming that the active source sends equally spaced packets. If the actual arrival time of a packet is not "too" early relative to the (TAT), in particular if the actual arrival time is after ($TAT-L$), then the packet is conforming; otherwise, it is non conforming. We have used this version of the GCRA as a principle in the design of the regulator's traffic shaping. The proposed regulator achieves the control by assigning each packet an Eligibility Time upon its arrival (ET) and by holding (the packet is actually blocked in a queue and hence prevents any further activity in the application) the packet till that time (ET). For our implementation model, we have used results proposed by a study in (Zhang and Ferrari, 1994) concerning a service discipline model for high speed networks switches.

Let (X_{min}) and (X_{ave}) be the minimum packet inter-arrival time and the minimum average packet inter-arrival time respectively. If we consider a regulator based on the (X_{min}, X_{ave}, I) set of parameters, the eligibility time of the k th packet, $ET(k)$, is defined according to the eligibility times of the preceding packets of the same connection by:

$$\begin{aligned}
 ET(k) &= -1 \quad \text{if } k \leq 0 \\
 ET(k) &= \text{Max}\left(ET(k-1) + X_{min}, I + ET\left(k - \frac{I}{X_{ave}}\right), AT(k)\right)
 \end{aligned} \tag{1}$$

Where:

I = The averaging interval over which (X_{ave}) is computed.

$AT(k)$ = Arrival time of packet (k) .

The (X_{ave}) is the minimum average packet inter-arrival time. It is computed according to an image of the packet arrival in the past. With the (X_{ave}) parameter, we enable the end-system to describe its future flow in greater details than just the peak rate (within the (X_{min})). With (1), we can smooth the source flow to the (X_{min}, X_{ave}, I) model and so, enhance resource utilization. This method chooses the maximum last cycle and the current busy period for averaging the current (ET) . The (X_{min}) parameter is computed according to the (PCR) , $(CDVT)$, (SCR) and (BT) parameters included in the traffic contract. Details on the computation of (X_{min}) are presented in the following sections.

Limitations of the GCRA in a multi-tasks operating system and motivation for a new scheme

The first regulator's implementation relied on the GCRA algorithm. We have noticed that it could hardly generate the required throughput and remarked a semi-permanent under-utilization of the reserved bandwidth leading to a performance degradation independently of the network behaviour. This conclusion was jointly made in (Radhakrishnan, et al., 1996). In fact, it has been shown that a stringent input rate control may unnecessarily increase the end-to-end delay by a significant amount which degrades the overall performances (Radhakrishnan, et al., 1996).

Moreover, the GCRA introduces access delays which can become prohibitive for the connections. Since each packet must wait until the arrival of its eligibility time, we had to use kernel blocking function. In the context of a multi-tasks operating system used for this implementation, we have noted that the granularity of the blocking function at the kernel level was inaccurate in certain situations. So, time was lost at each blocked packet arrival. Moreover, at each non-conforming packet, the connection passes into a *sleeping state* (due to the blocking function) waiting for a timer to expire. At the arrival of the eligibility time (expiration of the timer), it moves to the *runnable state*. Because of the context-switching, paging and swapping, a delay is introduced between the time the connection is ready to run and the time it is really scheduled as confirmed in (Khanna, et al., 1992). This delay becomes prohibitive for the transmission rate if the blocking frequency increases. The performance of this method was measured in different scenarios. The results are shown in Section 4.3.

In order to remedy to these problems and to reduce the access delays, a second approach was adopted. We have designed a traffic shaper conformant to a Leaky Bucket (LB) over long durations, allowing however periodic short burstiness.

The proposed scheme: A credit-based approach

The equivalence of the virtual scheduling and the leaky bucket algorithm was demonstrated in (Sathaye, 1996). We have adapted the definition of the leaky bucket to our implementation environment in order to make it less stringent and more flexible.

The original version of the leaky bucket algorithm $LB(I, L)$ can be viewed as a finite-capacity bucket whose real-valued content drains out at a continuous rate of 1 unit of content per time-unit and increases by (I) for each conforming packet arrival. If at a packet arrival, the content of the bucket is less than or equal to the limit value (L) , then the packet is conforming, otherwise, the packet is not conforming. The capacity of the bucket is upper bounded by $(L+I)$. Our approach is less stringent because it attributes an amount of tokens to the connection for a certain period (TT) . The credit-counter is at its maximum value at the beginning of the period. Each accepted packet consumes a token. When the bucket becomes empty, the whole connection is naturally blocked (since no packets are accepted) until the beginning of the next period. As mentioned before, this approach may introduce a burst effect within (TT) if packets inter-arrival time is very small. This effect can be reduced by reducing (TT) .

A credit amount is attributed to each connection. Its value depends on the source traffic type (CBR, VBR, ABR,...). In the next section, we will present the formula used to derive the credit expression in terms of the known ATM-Forum traffic management specifications. As the regulator is periodic, the attribution of the credit amount will be done at the beginning of each new period.

The credit calculation

The conformance to a specific leaky bucket $LB(I, L)$ can be controlled by maintaining an image of the credit counter. According to the traffic management specifications document (Sathaye, 1996), $N(t)$, the number of packets that can be emitted with spacing no less than (I) and still be in conformance with $LB(I, L)$ over any closed interval of length (t) is bounded by:

$$N(t) \leq \min \left(\left\lfloor 1 + \frac{t+L}{I} \right\rfloor, \left\lfloor 1 + \frac{t}{I} \right\rfloor \right) \quad (2)$$

In our approach, we limit the number of packets that can be sent over an interval (TT) but we don't control the spacing between 2 successive packets. We consider that the rate of a connection is defined over an averaging interval (TT) . It is determined by dividing the total number of bytes transmitted during this interval. A source can emit the maximum number of packets over a duration less than (TT) but can't exceed the maximum amount of allowed packets. It is blocked when its token bucket is empty until the beginning of the next period. The value of (TT) determines the allowed burstiness at a source. The larger this interval is, the higher the allowed burstiness will be. We have estimated (TT) by experimentation trials. A one second (TT) value was found to be optimal for the current platform. A more generic method is currently being studied for (TT) estimation. Relying on these considerations, the calculation of the credit for each connection is made according to these formula:

For VBR sources:

$$1) \text{ If } TT \geq MBS \times T \quad \text{then} \quad Credit = \left\lfloor 1 + \frac{TT + BT}{Ts} \right\rfloor \quad (3)$$

$$2) \text{ If } TT \geq MBS \times T \quad \text{then} \quad Credit = \left\lfloor 1 + \frac{TT}{T} \right\rfloor \quad (4)$$

Where:

BT = Burst Tolerance (in seconds).

$Ts = 1/(SCR)$ where (SCR) designates the Sustainable Cell Rate of the source.

MBS = Maximum Burst Size (in cells).

$T = 1/(PCR)$ where (PCR) is the Peak Cell Rate of the connection.

Equations (3) and (4) are conform to the ATM traffic management specifications (Sathaye, 1996).

For CBR sources:

Since our controller is implemented at the packet layer, a CBR source is considered as a VBR one at the TCP level. This is due to the fluctuations generated by the decomposition of a TCP packet into ATM cells. So, even if the (BT) and the (SCR) parameters are not defined for such connections within the traffic contract, we define them for our credit calculation. We use (3) and (4) to calculate the credit of such connections. The definition of the (SCR) and the (BT) parameters is explained in the following section.

The packet-to-cell parameter conversion:

The traffic shaping principles proposed by the credit-based approach conditions a *cell* input stream. The control is based on the traffic contract parameters expressed in terms of number of cells and their inter-arrival time throughout the life time of the connection. As we implement the controller in the TCP module, we have thought to adapt our control and the parameters used in the credit calculation to the control of the burstiness caused by the fragmentation of a TCP packet into ATM cells. The motivation is based on the fact that if a non conforming cell is dropped at the UNI, the whole packet is damaged and can not be reassembled by the destination host. We propose a solution which is based on the choice of the (BT) parameter. This parameter must be chosen so that fluctuations within the cellular traffic can be absorbed inside the TCP packet. Let (r) designate the TCP traffic rate (in bit/s) and (b) the burst tolerance (in bit) both corresponding to a packet scale leaky bucket parameters. Consider periodical arrival of TCP packets, i.e. one packet of variable length is generated each period (τ) . Packet scale conformance to a leaky bucket of parameters (r) and (b) is equivalent to :

$$A(m) \leq rm\tau + b \quad (5)$$

Where $A(m)$ is the amount of data generated during (m) consecutive packets. At the fluid scale, let $N(s,t)$ be the amount of data generated in the interval $[s,t]$. Conformance to a GCRA controller operating at the UPC involves $N(s,t)$. The GCRA can then detect violations even if the source remains conform to (5). This means that packet scale conform-

ance does not necessarily ensure cell-scale conformance if the same parameters of the leaky bucket are used. To avoid this situation, the choice of parameter (BT) used in the GCRA becomes important. The analysis that follows determines the (BT) and (SCR) parameters in function of (r) and (b) values.

Let $R(t)$ be the instantaneous TCP rate expressed in bits/s. The quantity of data emitted within an interval $[s, t]$ is defined by:

$$N(s, t) = \int_s^t R(t) dt \quad (6)$$

If (n) designates the smallest integer verifying $s \leq n\tau$ and (m) the highest integer that verifies $m\tau \leq t$, then we have:

$$s \leq n\tau \leq m\tau \leq t \quad (7)$$

$$n\tau - s \leq \tau \quad (8)$$

$$t - m\tau \leq \tau \quad (9)$$

And (6) becomes:

$$N(s, t) \leq \int_s^{n\tau} R(t) dt + \int_{n\tau}^{m\tau} R(t) dt + \int_{m\tau}^t R(t) dt$$

The expression of the middle designates $A(m-n)$ and is bounded by (5), we have:

$$N(s, t) \leq b + r\tau(m - n) + \int_s^{n\tau} R(t) dt + \int_{m\tau}^t R(t) dt$$

If (p) designates the peak rate of the source in bits/s, we have:

$$\int_s^{n\tau} (R(t) - p) dt \leq 0 \quad \text{and} \quad \int_{m\tau}^t (R(t) - p) dt \leq 0$$

which is equivalent to:

$$\int_s^{n\tau} R(t) dt \leq p(n\tau - s) \quad \text{and} \quad \int_{m\tau}^t R(t) dt \leq p(t - m\tau)$$

So, we have:

$$N(s,t) \leq p(n\tau - s) + p(t - m\tau) + r\tau(m - n) + b$$

Using (8) and (9), we obtain:

$$N(s,t) \leq r(t - s) + 2\tau(p - r) + b \quad (10)$$

With such an expression, we can conclude that the conformity at the packet level can be ensured if we choose:

$$BT = b + 2\tau(p - r) \text{ and } SCR = r$$

The parameter definition

In order to apply (3) and (4), we have to define for each type of source the parameters (*SCR*) and (*BT*). For VBR sources, these parameters are contract defined. For CBR sources, only the (*PCR*) and the Cell Delay Variation Tolerance (*CDVT*) parameters are defined. Since a CBR connection sends at the peak cell rate for all the connection time, then the (*BT*) parameter can be supposed equivalent to the (*CDVT*) one. Within the (*CDVT*) duration, we have a tolerance for cell clumping equal to :

$$BT = CDVT \times PCR$$

To define the (*SCR*) parameter for this type of sources, we have used this expression proposed in the ATM forum standard document (Sathaye, 1996):

$$MBS = 1 + \frac{BT \times SCR}{Ts - T} \quad \text{where} \quad Ts = \frac{1}{SCR} \quad \text{and} \quad T = \frac{1}{PCR}$$

The (*MBS*) parameter designates the maximum burst size. As CBR sources can emit at the (*PCR*) rate for all the connection time, we have transformed this parameter into the maximum number of bits that can be emitted within the interval (*TT*) (due to the tolerance admitted in the inter-arrival time). So we have:

$$Ts - T = \frac{BT \times SCR}{PCR \times TT - 1}$$

If we replace (*MBS*), (*Ts*) and (*T*) by their expressions as a function of (*SCR*) and (*PCR*), we have:

$$\frac{1}{SCR} = \frac{1}{PCR} + \frac{BT \times SCR}{TT \times PCR - 1}$$

Finally, we have:

$$SCR = \frac{PCR}{1 + \frac{BT \times SCR}{TT - \frac{1}{PCR}}}$$

To recapitulate:

* For CBR sources, let $(CDVT)$ and (PCR) be the 2 parameters specified in the traffic contract at establishment time. Since we consider the connection as bursty at the packet level, we suppose that we have 2 auxiliary parameters considered as a Burst Tolerance $(BT(c))$ and a Sustainable Cell Rate $(SCR(c))$ defined as follows:

$$BT(c) = CDVT \times PCR \quad \text{and} \quad SCR(c) = \frac{PCR}{1 + \frac{BT(c) \times SCR}{TT - \frac{1}{PCR}}}$$

To implement the control at the packet level, we have chosen for the LB algorithm $(LB(1/SCR(p), BT(p)))$ these parameters:

$$BT(p) = BT(c) - 2(p - SCR(p)) \quad \text{and} \quad SCR(p) = SCR(c)$$

* For VBR sources, let $BT(c)$ and $SCR(c)$ be the traffic parameters chosen by the connection which will be used by the UNI to perform the control. Then, we choose to implement the control at the packet level with $LB(1/SCR(p), BT(p))$ where $BT(p)$ and $SCR(p)$ are defined as follows:

$$BT(p) = BT(c) - 2(p - SCR(p)) \quad \text{and} \quad SCR(p) = SCR(c)$$

For the other types of traffic (ABR and UBR), we have left the calculation for a further study.

3.2 The scheduler's principle

The role of the scheduler becomes important if the system load increases so greatly that the endstation can not satisfy the whole needs of the connections. In such a situation, the packets are put onto the network with a certain delay because of the multiplexing of the different data flows. The scheduler's purpose is to introduce a set of parameters assigning 2 priority levels to the connections. The packets coming from the different connections are organized into 2 queues: a real-time queue and a non-real-time queue. For CBR and VBR connections which support real-time applications requiring tightly constrained delay variations, we steer the packets towards the real-time queue. For the ABR and UBR connections, we direct them to the non-real-time queue. The real-time queue has the highest priority. The scheduler services the packets using a non-preemptive FCFS discipline. The non-real-time packets are serviced only if there are no real-time packets. We have chosen the FCFS discipline because of its simplicity of implementation which is an important advantage in the context of high speed environment. The scheduler performance under heavy load conditions is subject to outgoing research.

4 IMPLEMENTATION

4.1 Environment

As mentioned before, a multi-tasks environment supports our implementation. It consists in a SOLARIS operating system running over a SPARC 20 station. SOLARIS is a system V Release 4 OS offering the Streams environment. TCP is implemented as a module within the ATM communication architecture. A stream consists in a full duplex connection made between a device driver monitoring the ATM card and the stream head supervising the interface with the user (Figure 2). For each connection established at the application level, a specific data structure is interpreted as the traffic contract of the source. These information elements are transferred by the stream head which is represented by the Transport Layer Interface (TLI) system routines down to the TCP layer.

Since a TCP connection corresponds to a kernel process in the system context, each source can be considered as a unique TCP process. We can then ensure the independence between the connections since we implement the control before the IP multiplexing. On the other hand, since the implementation is supported by a multi-tasks OS, we have to take into account the properties of this environment to ensure good performances of our controller. Within the SOLARIS operating system, the process scheduling leads to costly context switches (Khanna, et al., 1992). The TCP process depends on scheduling rules that are applied within the kernel and its performances are consequently affected by the background load supported by the system.

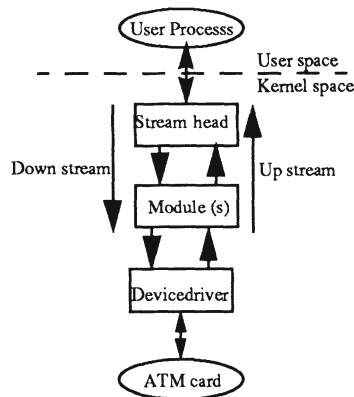


Figure 2 The streams architecture.

4.2 The controller's algorithm

The steps of the controller algorithm are:

1. At the first packet arrival, we calculate the credit allocated for the connection according to its type, to its traffic contract parameters and to the packet-to-cell parameter conversion principle. The credit consumed is initialized to zero. A timestamp indicates the beginning of a period.

2. At each packet arrival:

2.1. If a new period begins, the credit consumed is reinitialized to zero and the available credit is set to the maximum value calculated in the step (1). A new timestamp is relieved to indicate the beginning of another new period.

2.2. If sufficient credit is available for the packet, we consume an amount proportional to the packet size and the packet is passed down through the streams architecture.

2.3. If insufficient credit is available, we block the current process till the arrival of the next period. At the expiration of the timer, the process becomes runnable and the packet is sent down. A timestamp is relieved to indicate the beginning of a new period and the credit consumed is set to zero.

4.3 Results

A series of experiments has been made on a client-server architecture. All clients are supposed to have very long data transfer to a server. In order to compare the behaviour of the sources in the two cases when they are controlled and uncontrolled, we have activated two client processes simultaneously : the first supporting the control and the second without any rate constraints. In regulated mode, the client specifies its traffic contract via an interface and the parameters choosen are taken into account by the controller. The results of these experiments are reassembled in (Figure 3).

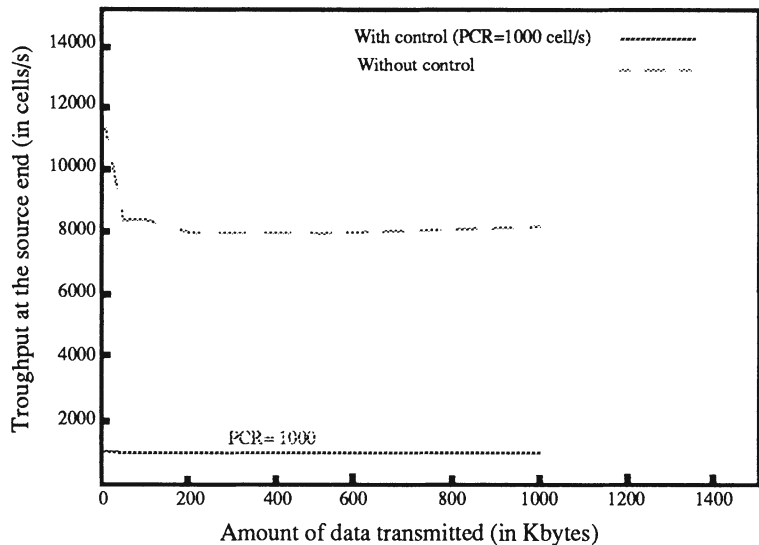


Figure 3 Controlled and uncontrolled behaviours.

A second experiment is shown in (Figure 4). It consists in comparing the performances of the GCRA algorithm with those of the proposed control algorithm. Since the GCRA is based on the control of the packet inter-arrival time, for each non conforming packet, the kernel blocking function is used to enforce the connection to reduce its transmission rate. We can notice in (Figure 4) that when only 2 sources are emitting simultaneously, the

performances of the 2 algorithms are nearly equivalent. When we multiply the system load by increasing the number of simultaneous connections, we note a degradation of the performances with the GCRA approach. This is due to the kernel scheduling and to the context switching. The blocking frequency is higher with GCRA. This leads to an excessive access delay which induces a performance drop. Since the operating system is not a real-time one, the accuracy has limits which is prohibitive in such cases.

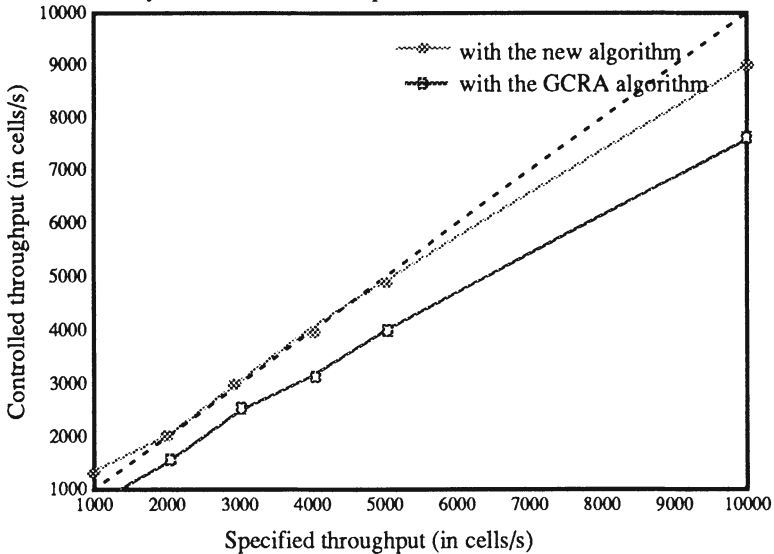


Figure 4 Comparision between GCRA and the new algorithm.

The curve in (Figure 5) shows the amount of time lost at each packet blocking depending on the number of active connections. For several connections, an amount of time which reaches the maximum value of 1.4 milliseconds is measured between the theoretical activation time of the process after a blocking caused by a non-conforming packet and the real activation time.

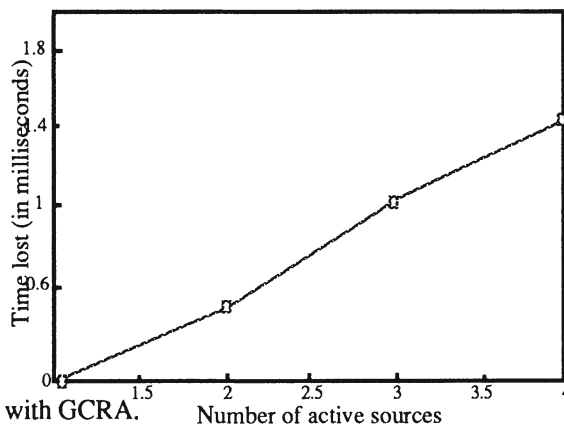


Figure 5 Time loss with GCRA.

So, we can notice in (Figure 6) the impact of the system load increase and the time loss on the bandwidth offered by the GCRA. We have activated simultaneously 4 sources having a bandwidth allocation of 5000 cell/s. Since the kernel blocking function becomes inaccurate, we remark a drop in the transmission rate with such a control.

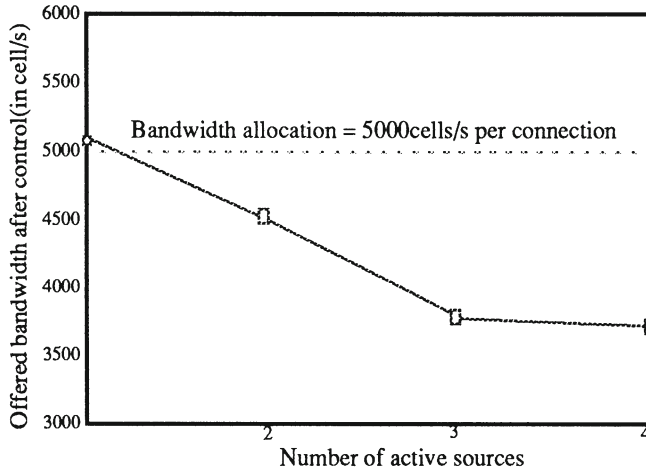


Figure 6 Bandwidth offered with GCRA.

5 CONCLUSION

In this paper, we have presented the design and the implementation of a traffic controller for applications using an IP over ATM environment. It achieves traffic shaping and connection scheduling at the source edges. A comparative study between the performances of the GCRA as described by the ITU and those of the proposed controller's is also delineated. We show the results obtained by both algorithms when implemented in a multi-tasks environment. A performance comparison shows that the stringency of the GCRA leads to excessive delays before data can be sent to the network. We show the adaptability of the proposed algorithm to the traffic control purpose in a multi-tasks environment. A quantitative means of making a packet-to-cell parameter conversion is explained. Such a conversion is needed since the implementation is done at the TCP level.

With this traffic shaping offered at the source edges, we can ensure guaranteed QoS for applications. The traffic shaper and the switches will collaborate to provide the required performances and good network channels utilization. We will investigate in a future work on the choice of the bandwidth allocation for each application in order to ensure ergonomic presentation (e.g WEB sessions) and to give the required performance guarantees. Resource sharing among users is also an important issue in the presence of a guaranteed network.

6 REFERENCES

- (Afifi, et al., 1996)
- Afifi, H., Bonjour, D., Elloumi, O., (May 1996). *TCP over Non Existing IP For ATM Networks- JECN'7 Budapest.*
- (Almesberger, 1996)
- Almesberger, W., (1996). *Arequipa: Design and Implementation*, ftp://lrcwww.epfl.ch/pub/arequipa/arequipa_d1-1.tar.gz, Technical Report 96/213. Nov 96
- (Marsan, et al., 1995)
- Marsan, A., Bianco, A., Lo Cigno, R., Munafo, M. (April 1995). *Shaping TCP Traffic in ATM Networks In proceedings ICT'95 Bali Indonesia.*
- (Cole, et al., 1995)
- Cole, R.G., Shur, D.H., Villamizar, C., (April 1995). *IP over ATM - A Framework Document* , Internet-Draft.
- (Elloumi, et al., 1995)
- Elloumi, O., Afifi, H., Rolin, P., Hamdi, M. (December 1995). *Issues in Improving TCP performance over ATM. In Proceedings of the IFIP ATM workshop on traffic management WATM'95 Paris.*
- (Garrett, 1996)
- Garrett, A.W., (May-June 1996). *A Service Architecture for ATM : From Applications to Scheduling. IEEE Network, May-June 1996.*
- (Jacobson, 1988)
- Jacobson, V., (1988). *Congestion Avoidance and Control. In the proceedings of SIGCOMM'88 Symposium, pages 314-32, Aug. 1988.*
- (Kandlur, et al., July 1995)
- Kandlur, D.D., Saha, D., Willebeek-LeMair, M., (July 1995). *Protocol Architecture for Multimedia Applications over ATM networks. CCR Volume 25 Number 3 July, 1995.*
- (Khanna, et al., 1992)
- Khanna, S., Sebrée, M., Zolnowsky, J., (1992). *Real-time Scheduling in SunOS 5.0. In Proceedings of Usenix 92-Winter 1992.*
- (Keshav and Hill, 1995)
- Keshav, S., Hill, M., (1995). *Semantics and Implementation of a Native-Mode ATM Protocol Stack.*
- (Laubach, 1994)
- Laubach, M., (1994). *Classical IP and ARP over ATM. RFC-1577, IETF 1994.*
- (Matthis, et al., 1996)
- Mathis, M., Mahdavi, J., (1996). *Forward Acknowledgement: Refining TCP Congestion Control. ACM Sigcomm 1996 Proceedings 281-291.*
- (Perloff and Reiss, 1995)
- Perloff, M., Reiss, K., (1995). *Improvements to TCP Performance in High Speed ATM Networks. Communications of the ACM. Feb 1995 Vol 38-2.*
- (Radhakrishnan, et al., 1996)

Radhakrishnan, S., Raghavan, S.V., Agrawala, A.K., (1996). *A flexible Traffic Shaper for High Speed Networks: Design and Comparative Study with Leaky Bucket - Computer Networks and ISDN systems* 28 (1996) 453-469.

•(Radhakrishnan and Raghavan, 1994)

Radhakrishnan, S., Raghavan, S.V., (1994). *Network Support for Distributed Multimedia - Issues and Trends, SEACOMM' 94.*

•(Sathaye, 1996)

Sathaye, S., (1996). *The ATM forum Traffic Management Specification, version 4.0 ATM Forum/95-0013R6 April 96.*

•(Zhang and Ferrari, 1994)

Zhang, H., Ferrari, D., (1994). *Rate-Controlled Service Disciplines. Journal of high speed networks* 3 (1994), 389-412.

•(IP-over-ATM WG, 1995)

IP over ATM WG (1995). *Recommendations for the ATM Forum's Multiprotocol BOF.*

PART FOUR

Quality of Service

On Routing with QOS Constraints in ATM Networks

Dirceu Cavendish and Mario Gerla

University of California at Los Angeles

*Computer Science Dept., School of Engineering and Applied Science,
405 Hilgard Avenue, Los Angeles, CA, 90024, USA.*

Telephone: 1 310 8254367. Fax: 1 310 8257578.

email: {dirceu,gerla}@cs.ucla.edu

Abstract

Multimedia applications require guaranteed QOS, which will be provided by multimedia networks, such as ATM, by resource reservation. The complexity of routing and allocating resources for ATM applications vary according with applications end-to-end constraints, and essentially range from P to NP complete problems (Wang et al (1995)). In this paper, we expand the class of known P type, constrained routing problems, by including two applications of significance in ATM, namely loss sensitive ABR traffic, and delay sensitive VBR traffic. For such routing problems, we present a $O(N^4)$ polynomial time algorithm, prove its correctness, and illustrate its effectiveness with simulation experiments.

Keywords

ATM Routing, End-to-end Flow Control, Quality of Service Constraints

1 INTRODUCTION

Broadband networks based on Asynchronous Transfer Mode (ATM) are being designed for the integration of various types of traffic, such as voice, data, and video. These networks are virtual circuit oriented, and they must provide end-to-end Quality Of Service (QOS) guarantees to the connections, which may vary from application to application.

The goal of providing end-to-end guarantees has a considerable impact on the design of algorithms at various protocol levels of the network. To mention a few, intermediate switches must provide bounds on service delays and loss; Call Admission Control (CAC) must take into account the available trunk bandwidth resources so that the end-to-end requirements of existing connections be maintained; Routing algorithms must provide routes which conforms with users end-to-end requirements while minimizing the cost of a connection. The design of efficient scheduling algorithms for meeting QOS requirements is an active research topic (Georgiades et al (1996), Ling et al (1996), Parekh et al (1993, 1994)).

Routing with bandwidth allocation has been considered in Wand et al (1995), and is essentially a polynomial solvable problem with the same time complexity as Dijkstra algorithm. Routing problems which consider pairs of link costs of additive nature, such as delay and number of hops, have been studied as restricted shortest path problems, as in Hassin (1992), and are known to be NP-complete problems. The complexity of these problems depends heavily on the definition of link costs and end-to-end requirements.

In ATM nomenclature, links can be associated with two types of parameters (PNNI Draft Specification (1994)), link metrics and link attributes. Link **attributes** are parameters which can be consulted in isolation when deciding the link eligibility for carrying a connection. Bandwidth is an example of a link attribute. Link **metrics**, on the other hand, are of additive nature, hence they need to be considered in conjunction with other links in a given path in order to decide path eligibility for carrying a QOS constrained connection. Delay is an example of a link metric. When link attributes only are involved, the problem has been shown to be easily solvable (Wang et al (1995)). For instance, the available bandwidth of a path is the minimum bandwidth available among all its links. However, for routing problems involving multiple link metrics, such as delay and network operation cost (called administrative cost in PNNI Draft Specification (1994)), the problem becomes NP-complete.

This paper identifies, for the first time, a new class of polynomially solvable routing problems, which in terms of complexity lies between routing problems considered by Wang et al (1995) and Hassin (1992). We argue that the task of routing some ATM applications can be modeled by this class of problems.

The paper is organized as follows. Section 2 defines precisely the routing problems of our interest, including the class introduced in this paper. In section 3, we present an algorithm for routing this new class of problems, based on executing rounds of the Dijkstra shortest path algorithm, and prove its correctness. Section 4 identifies ATM applications for which our routing algorithm is suitable. In section 5 we describe a network topology used to evaluate routing for some of the proposed applications. We present simulation results, comparing call acceptance probability versus network load for these applications. Conclusions are drawn in the last section. An Appendix is included with pseudo code of the routing algorithms used in the simulation experiments.

2 ROUTING PROBLEM DEFINITION

As mentioned earlier, multimedia networks will require routing algorithms which are capable of dealing with multiple link metrics. In this section we start by describing a NP-complete routing problem precisely. Then we define a routing problem of lesser complexity.

We assume that each link has only two independent metrics. We model a multimedia network as a generic graph, in which paths must be established between nodes so that path constraints be met. Thus, a graph $G = (V, E)$ consists of a set V of vertices and a set E of edges. Associated with each edge $e \in E$ is a length $l(e) \in R^+$ and a weight $w(e) \in R^+$. A path p is a sequence of distinct vertices v_1, v_2, \dots, v_k such that $(v_i, v_{i+1}) \in E$ for $i = 1, \dots, k-1$. The set (v_i, v_{i+1}) is called the edges of p . The length (weight) of p , $L(p)$ ($W(p)$) is the sum of the lengths (weights) of the edges of p .

The NP-complete problem is defined as follows: find a minimum length path connecting two specific vertices (s, d) , while satisfying a path weight bound W . This routing problem

can be formulated by the following optimization problem: Minimizing the cost function $l(p)$ of a path p between a specific source/destination pair (s,d) , where

$$l(p) = \begin{cases} \sum_{i \in p} l(i) & \text{if } \sum_{i \in p} w(i) < W \\ \infty & \text{otherwise} \end{cases}$$

As mentioned earlier, this routing problem has been studied in Hassin (1992). We now define a new routing problem. Let the cost of a path be defined slightly differently, as follows:

$$l(p) = \begin{cases} \sum l(v_i, v_{i+1}) & \text{if for each edge } (v_i, v_{i+1}), w(v_i, v_{i+1}) > M \sum l(v_i, v_{i+1}) \\ \infty & \text{otherwise} \end{cases} \quad (1)$$

where M is a constant. Notice that in this definition the second link parameter w can be considered a link attribute, in a sense that one does not need the values of other w links parameters to decide link eligibility. However, one still needs to consult the l metric values of the other links to decide eligibility. We define the shortest path length from s to d to be:

$$\delta(s, d) = \begin{cases} \min[l(p) : s \rightsquigarrow d] & \text{if } \exists \text{ a path from } s \text{ to } d \\ \infty & \text{otherwise} \end{cases} \quad (2)$$

In the next section, we present a routing algorithm which finds shortest paths according to this last definition between any two vertices of a graph.

3 BOTTLENECK DISCARDING ROUTING PRINCIPLE

The main routing strategy is rather straightforward. We use a Dijkstra algorithm to find the shortest path between a given pair (s,d) of vertices. However, at each step of the algorithm, when adding new edges, we verify that the cumulative path weight does not violate the weight constraint defined in Eq. (1), which would raise the cost of the path to infinity. If the constraint is violated, the *bottleneck* link, which is defined to be the link with highest weight value so far, is discarded, and a new attempt to find a feasible path starts. We define a phase of the algorithm to be the interval from when the algorithm starts path computation until it either finishes the computation or it detects constraint violation. The algorithm halts when either it finds a path or the source gets disconnected, in which case no path is found. We name this routing strategy the bottleneck Dijkstra algorithm, B-Dijkstra. A B-Dijkstra pseudo-code follows.

3.1 B-Dijkstra Pseudo-code

B-DIJKSTRA $G(V,E)$

B-Initialize(G,s);

$S \leftarrow \{\}, Q \leftarrow V[G]$;

while $Q \neq \emptyset$ **do**

If (B-Extract-Min(Q, u, l) = *TRUE*)

$S \leftarrow S \cup \{u\}$;

If $u = t$ **found**;

For each node $v \in Adj[u]$ **do**

 B-Relax(u, v, l_{uv}, w_{uv});

else

 B-Initialize(G,s);

where the subroutines B-Initialize(G,s), B-Extract-Min(Q,u,l) and B-Relax(u, v, l_{uv}, w_{uv}) are described below:

B-Initialize(G,s)

For each node $v \in V[G]$

Do $l_v \leftarrow \infty$; $w_v \leftarrow \infty$; $\pi_v = NIL$;

$l_s \leftarrow 0$; $w_s \leftarrow 0$;

$w_{max}^i = \infty$; $bl^i = NIL$;

B-Extract-Min(Q,u,l)

$d_{min} \leftarrow \infty$; $u \leftarrow NIL$;

For $i \in Q$ **do**

If ($l_i < d_{min}$)

 { $d_{min} \leftarrow l_i$; }

$u \leftarrow i$;

If ($w_u < W_{st}$)

RETURN *TRUE*;

else

$l_{*\pi(bl^u),bl^u} \leftarrow \infty$;

RETURN *FALSE*;

B-Relax(u, v, l_{uv}, w_{uv})

If ($l_v > l_u + l_{uv}$)

If ($w_{max}^u > w_{uv}$)

 { $w_{max}^v = w_{max}^u$; $bl^v = u$; }

else

 { $w_{max}^v = w_{uv}$; $bl^v = v$; }

$l_v \leftarrow l_u + l_{uv}$;

$w_v \leftarrow w_u + w_{uv}$;

S is a set of nodes whose current shortest path is maintained, and Q is a priority queue with nodes in $V - S$ with their current distances l_i . Each node u has a pointer π_u to its

previous node in the current shortest path, which is initially set to NIL. B-Extract-Min fetches the node outside S which is closest to the source, and B-Relax updates nodes distances to the shortest ones, as is standard in Dijkstra's algorithm. The twist here is that, by keeping track of the path weight $w(p)$, B-Extract-Min routine checks if the path weight constraint is violated. If this happens, the bottleneck link (in weight) is discarded and the algorithm restarts. Since for each Dijkstra phase of the algorithm, at worst the link next to the source is discarded when we are next to reaching the destination t , we have $|E|$ phases of the usual Dijkstra algorithm at the most. Each Dijkstra phase taking $O(N^2)$ time (Cormen et al (1990)), for a complete graph we obtain $O(N^4)$ complexity for B-Dijkstra. Although this time seems to be still excessive for large networks, simulation experiments reveal that the average running time of B-Dijkstra is very close to the original Dijkstra, mainly because we are likely to discard only few links in the computation of a path. In the following subsection, we prove the algorithm correctness.

3.2 B-Dijkstra Correctness Proof

We now prove B-Dijkstra correctness. More formally:

Theorem 1 *B-Dijkstra algorithm computes the (s, t) shortest path as defined in definitions 2 and 1.*

To prove the theorem, we first need the following lemma:

Lemma 1 *Consider a generic node y at an intermediate step of the s, t shortest path computation. Then, $l(s, y) = \delta(s, y)$ when node y is inserted into set S , after being chosen by B-Extract-Min.*

Proof. The proof is identical to the equivalent claim for DIJKSTRA algorithm, which can be found in Cormen et al (1990), for example. The additional path constraint does not invalidate the proof there described. \square

Lemma 1 guarantees that, as long as edges are not discarded, the shortest distance from s to any destination is obtained when the algorithm halts. Next, it remains to be proven that the discarding of the bottleneck edge does not invalidate the lemma.

Figure 1 depicts a typical step of the algorithm. Let node u be the node fetched by B-Extract-Min, when computing the optimal path from source s to destination t . Let also edge (m, n) be the bottleneck edge of the path (s, u) . Moreover, define the bottleneck resource $w^b(u)$ as the minimum value $w(v_i, v_{i+1})$ along the path (s, u) .

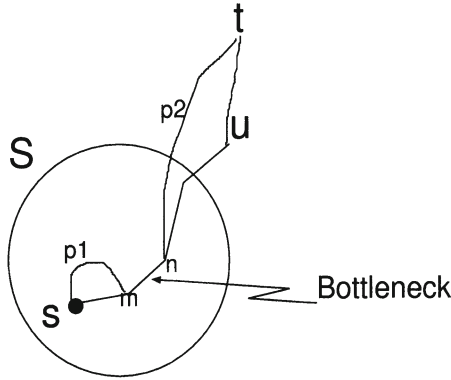


Figure 1 B-Dijkstra correctness proof

Proof. For each node u fetched by B-Extract-Min, $u \in Q$, and $\delta_u = \min_{i \in Q} [l_i]$, two cases are possible:

- i) $w^b(u) > M \sum_p l(v_i, v_{i+1})$: In this case, lemma 1 guarantees minimum distance computation.
- ii) $w^b(u) < M \sum_p l(v_i, v_{i+1})$: In this case, the bottleneck edge is eliminated. We must show that no feasible path containing such bottleneck exists. Let's suppose there is a feasible path $(s, p1, m, n, p2)$. Notice that edge (m, n) may or may not be the bottleneck of this path. Let the new path bottleneck value at t be $w^{b'}(t)$. We know that $w^b(u) < Ml(s, u)$. By using lemma 1 for nodes m and u , we know that $l(p1) \geq \delta(s, m)$, and thus the path $(s, p1, m, n, p2)$ is longer than $l(s, u) = \delta(s, u)$. If the bottleneck edge for path $(p1, n)$ is (m, n) , the previous observation that the alternate path has longer length than the original one leads to the conclusion that the path constraint is also violated for the alternate path. If the bottleneck of path $(p1, n)$ is not (m, n) , $w^{b'}(n)$ has an even smaller value than the original bottleneck $w^b(t)$, which causes the path constraint to be violated as well. Therefore, there is no feasible alternate path that includes bottleneck (m, n) .

□

4 ROUTING ATM TRAFFIC WITH B-DIJKSTRA

In this section, we describe various ATM networks applications in which the B-Dijkstra can be effectively used as a routing algorithm. First we review the key network assumptions. ATM switches are output buffered. The resources to be allocated in the switches in order to guarantee end-to-end requirements are buffer space and service rate (bandwidth). Although commercial ATM switches may not allow buffer allocation, recent research papers indicate that buffer allocation is necessary for QOS control (Georgiadis et al (1996), Mascolo et al (1996)). Each switch has a dedicated buffer space per output port (although this assumption can be relaxed). We assume route computation at the source (as per link state, or OSPF routing). Available link resources (buffer, bandwidth) are broadcasted at

regular intervals to each ATM node, where a topology map of the network is kept. In this map, each link (i, j) is associated with a buffer size B_{ij} and a Capacity (bandwidth) C_{ij} . Resources are reserved in a FCFS fashion at call request time. If there are N_{ij} connections sharing link (i, j) , where connection k has (b_{ij}^k, c_{ij}^k) buffer and capacity resources allocated, respectively, the broadcast information for that link is given by:

$$\begin{aligned} \text{available capacity: } c_{ij}^{av} &= C_{ij} - \sum_1^{N_{ij}} c_{ij}^k \\ \text{available buffer: } b_{ij}^{av} &= B_{ij} - \sum_1^{N_{ij}} b_{ij}^k \end{aligned}$$

We do not consider failures of any type (neither link nor switch) in this paper.

4.1 Routing Loss sensitive ABR traffic

Loss sensitive ABR traffic is a data application characterized by some minimum bandwidth requirement, MCR. The ATM Forum, in its PNNI Draft Specification (1994), allows for loss sensitive ABR traffic. TCP traffic carried through an ATM network is an example, where cell loss triggers retransmission at TCP level, leading to throughput degradation. Traffic shaping may be performed at the network entry point to regulate source transmission rate. Within the network, buffers as well as bandwidth must be reserved to keep cell loss at an acceptable level.

For this type of application, the sustainable input rate is clearly related to buffers available at each link traversed by that connection, level of acceptable loss, and feedback loop delay, which in turn depends on the chosen path. For a connection (s, t) requesting (b_{st}, c_{st}) resources, the routing algorithm must find the minhop path which includes only links (i, j) such that $c_{ij}^{av} > c_{st}$ and $b_{ij}^{av} > b_{st}$. To complicate matters, however, while c_{st} comes directly from traffic declaration, b_{st} is likely to depend upon the round trip delay of the path yet to be computed, since the performance of the congestion control mechanism is affected by the round trip delay. For example, if the rate controller described in Mascolo et al (1996) is used, and zero cell loss is required, the buffer b_{st} to be reserved for a connection (s, t) is:

$$b_{st} = c_{stmax} \left(\frac{1}{K_c} + RTD \right)$$

where $K_c > 0$ is the feedback control gain, c_{stmax} is the maximum input rate of connection (s, t) , and RTD is the round trip delay of the connection. This simply states that we need buffers of the order of the number of cells circulating inside the feedback loop. For sake of generality, by choosing the fastest rate controller ($K_c = \infty$), we adopt the simplified relation:

$$b_{st} = K c_{stmax} RTD \tag{3}$$

where K is a proportionality constant to be tuned according to the acceptable cell loss level. We conjecture that this relation must hold regardless of the rate control scheme

used, as long as the controller attempts to set a limit on cell loss, and therefore takes into account the round trip delay. Of course the minimum RTD path exhibits the smallest buffer requirement. A simple approach would be to use a Dijkstra algorithm to compute the minimum RTD path, and then check if the following constraint is satisfied for each link (i, j) along the chosen path:

$$b_{ij} > K c_{stmax} RTD \quad (4)$$

However, if repeated call requests occur between a given source/destination pair, this approach would eventually lead to call rejection due to exhaustion of buffer resources, even though an alternate path with large buffer space might exist.

Performance can be improved using a QOS routing algorithm as follows. Let us define the propagation delay of link i , pd_{ij} , in terms of the number of inflight cells in the link (i.e. $pd_{ij} = prop.delay_{ij} \times c_{ij}$). Furthermore, since for ABR connection (s, t) only the minimum rate c_{st} must be guaranteed by the network, we rewrite equation (3) as $b_{st} = K c_{st} RTD$ by adjusting the gain K accordingly. To support the feedback control, we assume that the links (i, j) are bidirectional, i.e. $pd_{ij} = pd_{ji}$. The pseudo-code of the algorithm for finding a path for ABR connection (s, t) with minimum rate c_{st} is provided in the Appendix.

The ABR-DIJKSTRA algorithm follows the same relaxation principle as B-Dijkstra, with bottleneck discarding strategy. The minor differences are: At initialization, ABR-Dijkstra first discards all links which do not have the minimum required capacity. It also sets the available buffer space for a link to the minimum between the forward and reverse directions, since buffers are reserved in both directions to establish the feedback loop. For each node v , there is a variable b_{min}^v that records the **minimum** buffer space available among all nodes in the shortest path from s to v . Initially, $b_{min}^v \leftarrow \infty$. There is also the variable bnk which keeps track of the location where such minimum occurs (i.e. the **bottleneck link**). This way we do not have to recheck equation (4) for each node every time a path distance is changed.

4.2 Routing VBR traffic with bounded delays

The most common example of VBR traffic is compressed video, for which bounds on end-to-end delays must be guaranteed. Video applications are expected to specify burstiness and average bandwidth. In turn, switches must allocate bandwidth and buffers necessary to cope with the burstiness and bandwidth declared, delivering the required delay performance (Georgiades et al (1996), Parekh et al (1993,1994)).

VBR routing requires the computation of bounds on end-to-end delay for all feasible paths. Among these, the one with minimum administrative path cost should be elected as the preferred one, in order to maximize network resource utilization. The end-to-end delay consists of two components: propagation delay; switching and scheduling delay. It turns out that propagation delay is additive and renders the problem NP complete. Switching and scheduling delays are also additive, but can be negotiated on a per link basis (as shown below), and therefore can be handled with a P problem formulation of the type shown in Eq. (1). In our approach, we ignore link propagation delays, assuming them negligible in face of switching and scheduling delays. In case propagation delays are significant, the bottleneck discarding strategy could be used for path computation in this

case also, although there is no guarantee that the resulting path is optimum. In any event, a routing strategy which considers both types of link delays should take into account the delay negotiation presented in this section.

Scheduling disciplines have been proposed recently for providing switch delay bounds for multimedia connections (Georgiades et al (1996), Parekh et al (1993,1994)). We use the work of Georgiades et al (1996) to express the dependencies between end-to-end delay bounds, bandwidth, burstiness, and link buffer and bandwidth allocation. In their work, end-to-end delay bounds are guaranteed through hop-by-hop traffic shaping, so that traffic burstiness be kept on the same levels throughout the connection path. For a call (s, t) requiring end-to-end delay bound of W_{st} , given its source traffic shaper, which can be a leaky bucket with parameters (σ, ρ) , we can compute the maximum delay w_{ij} incurred at a link (i, j) , with rate and buffer resources c_{ij}^{av}, b_{ij}^{av} respectively. Notice that σ specifies the maximum burstiness of the VBR connection, whereas ρ represents its average bit rate. More precisely, traffic shaping and scheduling algorithms will guarantee bounds on waiting and service delays provided that for output link (i, j) we have $b_{ij}^{av} > \sigma, c_{ij}^{av} > \rho$. Namely, using a scheduling policy, such as NPEDF, the switching and scheduling delay bound w_{ij} can be specified as:

$$w_{ij} = \frac{\sigma}{\rho} + \frac{\sigma}{c_{ij}} \quad (5)$$

where c_{ij} is the total capacity of output link (i, j) . Disregarding propagation delays, a bound on end-to-end delay is obtained as follows:

$$w_{st} = \sum_1^M w_{ij} \quad (6)$$

For networks with non uniform link capacities, equation (5) tells us that different switches offer different delay bounds. For now let us assume a uniform link speed c . Thus, the second factor of equation (5) is the same on all links of a given connection, and its end-to-end delay contribution is minimized by choosing the minimum hop path. Notice that link delay w_{ij} can be reduced **for each node along the path** by allocating bandwidth ρ' larger than the declared rate ρ .

We summarize these observations in the following way: given a VBR connection (s, t) with parameters (σ, ρ) and end-to-end delay bound W_{st} , a path p with M hops is said to be eligible if we can reserve for each link (i, j) , $b_{ij} = \sigma$, and $c_{ij} = \rho$. Moreover, if $\sum_1^M w_{ij} < W_{st}$ we are done. Otherwise, we must renegotiate the bandwidth $\rho' > \rho$ so that we reduce the end-to-end delay to the required level. In this case, from Eq. (5), we have:

$$\rho' > \frac{M\sigma}{W_{st} + \frac{M\sigma}{c}} \approx \frac{M\sigma}{W_{st}} \quad (7)$$

The approximation made in the last equation is used to argue in favor of using the bottleneck discard strategy only. Namely, Eq. (7) resembles equation (4), which states

buffer requirements for ABR traffic. If RTD is measured in number of hops, these two relations are identical. The path delay, however, is still computed using the exact equation. The pseudo code for the routing strategy is presented in the Appendix.

The algorithm computes the minimum hop path from s to t , checking at each step (when a new link is added) if equation (7) is satisfied. To do that, similarly to the ABR traffic algorithm, the position of the bottleneck link is tracked. If the minimum hop path can not satisfy equation (7), the bottleneck link is discarded, and a new instance of the algorithm is started.

Notice that the proposed algorithm will compute a minimum hop path which provides the tightest lower bound on the end-to-end delay required by the connection. This way, it is reasonable to expect that network resources are maximized, which leads to high overall network throughput.

5 SIMULATION EXPERIMENTS

In this section, we illustrate how the bottleneck based Dijkstra routing algorithm provides better network utilization by comparing it with the usual shortest path (minimum hop) and a maximum capacity algorithms. Our figure of merit here is the call acceptance probability for a given network load.

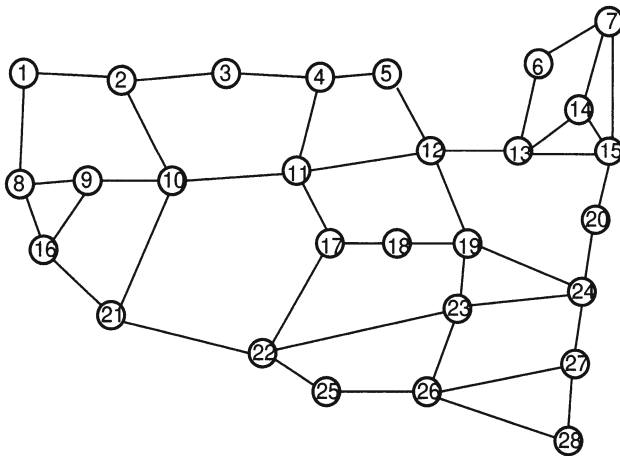


Figure 2 Network topology

We have simulated at call level a continental US size network. Figure 2 shows the network topology used in our experiments. The network diameter is 7, and the average node degree is 3.07. Link bandwidth is set to 155Mb/s. Buffers are provided on a per output link basis, with a uniform size of 155Mbits, or approximately 365K ATM cells. Since the network is of a small size, hierarchical routing, as specified by PNNI Draft Specification (1994), is not necessary. However, the routing strategy can be applied to each level of a more general hierarchical routing algorithm. A flooding scheme is used to

convey link state information to all nodes. However, link state messages are generated in a periodic fashion only, not on an event driven basis. This is so because generally the effectiveness of event driven link state dissemination depends on the so called significant change events in PNNI nomenclature, or events that change network topology, such as link failures and the like. Since we do not include events of such nature, a periodic flooding strategy suffices. A new flooding procedure is initiated approximately every 50ms, so that source nodes have an accurate view of the network state. Call arrivals are Poisson distributed. Each source randomly selects a destination to place a call to, and uses a specific routing algorithm to compute the path. If no path is found, the call is rejected at the source. Otherwise, a call request is placed, which can be followed by a call acceptance, in case the appropriate resources are available, or call rejection, in case the resources are no longer available (i.e. the state of the resources along the path has changed with respect to the information on which call acceptance at the source was based upon).

Switching delays case study

In this set of simulation experiments, calls are placed at a rate of 200 per second per node. A VBR call is characterized by: burst duration of 70 msec ($\sigma = 165$ cells), average rate $\rho = 10$ Mb/s, and maximum end-to-end delay (excluding propagation delays) $W = 35$ msec. Three routing strategies are compared: pure minhop Dijkstra routing; maximum capacity Dijkstra routing; and VBRSW-Dijkstra routing. The results are shown in Figure 3.

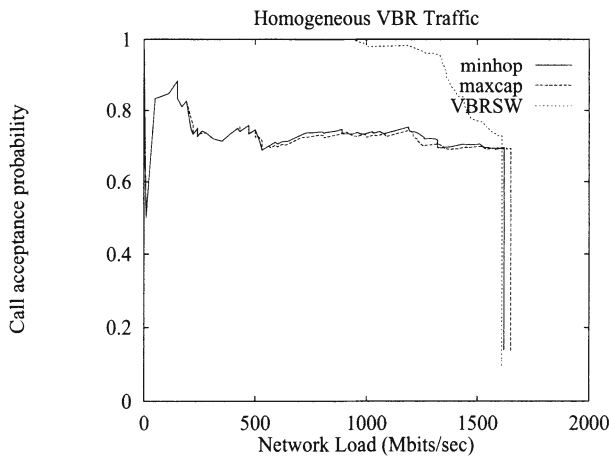


Figure 3 Call acceptance probability for homogeneous VBR traffic with switching and scheduling delays only

As expected, VBRSW-Dijkstra outperforms both maximum available capacity and minimum hop routing over the entire feasible network load range, up to saturation. Surprisingly, the two other schemes lead to similar call acceptance results. However, the exami-

nation of simulation traces of the routing process shows that, although both algorithms exhibit similar behavior, they differ greatly in the routes computed. When the network is lightly loaded, both algorithms choose the same paths, since the available capacity is the same for all links. However, as network load grows, the minhop strategy tends to select short but saturated paths, which lead to call rejection due to end-to-end constraint violation. In turn, the maximum capacity algorithm tends to select less congested, but longer, paths, which also cause end-to-end constraint violation. VBRSW-Dijkstra performs best among the three for all network loads because it keeps track of the minimum hop routing strategy among all feasible paths in terms of end-to-end delay requirement. Thus, it always provides the shortest possible path on which the delay QOS requirement is met. Furthermore, traces have shown that all successful paths found by minhop are also found by VBRSW-Dijkstra. The latter finds some additional (longer) ones, not explored by the minhop algorithm. By exploring alternative paths, VBRSW-Dijkstra is able to accommodate more calls over a broad network load range. We find that bottleneck link discarding happens only once or twice during a path computation, and hence the average running time of VBRSW-Dijkstra is close to the original Dijkstra algorithm, $O(N^2)$. This applies to other experiments, as well. Of course we should expect this number to increase for dense networks.

Rate Controlled ABR case study

Here we assume a network with available capacity for ABR traffic equal to 50Mb/sec on each trunk. Call arrival process is still Poisson, with an average of 20 calls/sec per network node. Each call requires 1Mb/sec rate, and zero cell loss. Thus, a feasible path must satisfy the buffer requirement $b_{ij} > c_{stmax}RTD$ for each link (i, j) along the path. Figure 4 shows the results of minhop, maxcap, and ABR-Dijkstra routing algorithms.

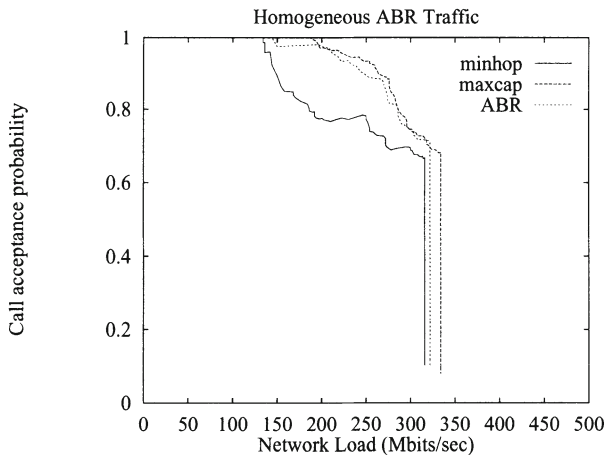


Figure 4 Call acceptance probability for homogeneous ABR traffic

We see that pure minhop routing performs poorly. An interesting and rather surprising result is that in this case maximum capacity routing and ABR-Dijkstra routing perform similarly. In fact, maxcap seems to even outperform ABR-Dijkstra. This is surprising because ABR-Dijkstra was proven optimal in this problem, and hence should be able to find a feasible path whenever maxcap finds one. Therefore, ABR-Dijkstra call acceptance should be at least as high as for maxcap. The explanation of this apparent discrepancy is based on two facts. First, for our particular set of parameters, buffer availability is closely coupled with available link capacity, so that the buffer constraint is automatically satisfied by the maximum capacity routing strategy. In fact, calls are uniformly placed throughout the network, and hence the higher the load is, the lower the buffer space. Thus, the maximum capacity path is very likely to satisfy also the buffer constraint, since link delays are small, and hence few extra links on a path do not impact significantly on buffer requirement. Notice that, although the end-to-end constraint here is similar to the one tackled by VBR-SW-Dijkstra, in the previous case few extra links would easily violate the end-to-end constraint, yielding to poor maxcap performance. Secondly, the route computation is based on link state information which is outdated with respect to the actual state of the network resources found during call set up time *. Notice that the ABR-Dijkstra solution is generally “tight”, i.e. it has little slack in terms of buffers and bandwidth. Thus, a minor change in link resources along the chosen path may render the path unfeasible. The maxcap path, on the other hand, is more robust to such changes, since it has the path with the largest possible buffer/bandwidth slack.

6 CONCLUSION AND FUTURE WORK

We have proposed a new polynomial routing algorithm, called B-Dijkstra (Bottleneck-Dijkstra), for ATM routing with path constraints. Although the worst case analysis of B-Dijkstra shows $O(N^4)$ complexity, simulations have shown that in practice the average running time of B-Dijkstra is close to $O(N^2)$, i.e. the complexity of the original Dijkstra algorithm. We have demonstrated the use of the B-Dijkstra algorithm in ATM networks for various traffic types, using recent research results regarding congestion control of ABR traffic and scheduling and traffic shaping of VBR traffic.

Simulation experiments based on homogeneous scenarios illustrate the concepts involved in routing end-to-end QOS constrained traffic. Although the overall behavior of the proposed routing algorithms is well described by the simulation results, the system under study was highly dynamic due to allocation and deallocation of resources, link state broadcasting, and propagation delays. Thus, a more comprehensive set of simulations would be useful to understand how much stale link information affects the performance of QOS routing algorithms. Also, it would be important to investigate whether it pays off to have different routing strategies for different applications in a network with heterogeneous traffic.

The algorithm implementation is assumed to be decentralized (or link state). That is, each source has a global picture of the network. This is in accord with PNNI Draft Specification (1994). However, routes are computed at call arrival instants. In case route computation time is large for a particular call admission procedure, caching routes for

*This phenomenon is more accentuated for light network loads, when more connections are being placed.

various traffic classes may be an alternative. This, however, may aggravate the “stale” routing problem. How this scheme affects resource reservation will be subject for future research.

Finally, the routing of VBR and ABR traffic for multicast connections is an issue of increasing importance given that many ATM applications are of multicast nature. We are currently investigating schemes for routing one-to-many, as well as many-to-many connections with QoS constraints.

REFERENCES

- Georgiadis, L., Guerin, R. and Peris, V. (1996) Efficient Network QoS Provisioning Based on per Node Traffic Shaping. *INFOCOM96*, vol.1, 102-110.
- Ling, T. and Shroff, N. (1996) Scheduling Real-Time Traffic in ATM Networks. *INFOCOM96*, vol.1, 198-205.
- Mascolo, S., Cavendish, D. and Gerla, M. (1996) ATM Rate Based Congestion Control Using a Smith Predictor: an EPRCA Implementation. *INFOCOM'96*, vol.2, 569-576.
- Wang, Z. and Crowcroft, J. (1995) Bandwidth-Delay Based Routing Algorithms. *GLOBECOM'95*, 2129-2133.
- Various contributors (1994) PNNI Draft Specification. *ATM Forum 94-0471R9*.
- Parekh, A. K. and Gallager, R. (1994) A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple Node Case. *IEEE/ACM Transactions on Networking*, Vol. 2, No. 2, 137-150.
- Parekh, A. K. and Gallager, R. (1993) A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case. *IEEE/ACM Transactions on Networking*, Vol. 1, No. 3, 344-357.
- Hassin, R. (1992) Approximation Schemes for the Restricted Shortest Path Problem. *Mathematics of Operations Research*, vol. 17, no.1, 36-42.
- Cormen, Leiserson and Rivest (1990) Introduction to Algorithms, *McGraw-Hill*.

ACKNOWLEDGEMENT

This work has been supported by CNPq Grant 201597/93-4 and by NSF Grant NCR-9305376

7 APPENDIX: ROUTING ALGORITHMS PSEUDO-CODE

ABR routing algorithm

ABR-DIJKSTRA $G(V,E)$

```

ABR-Initialize( $G,s$ );
 $S \leftarrow \{s\}$ ,  $Q \leftarrow V[G]$ ;
while  $Q \neq \emptyset$  do
    If (ABR-Extract-Min( $Q,u,d$ ) = TRUE)
         $S \leftarrow S \cup \{u\}$ ;
        If  $u = t$  found;
        For each node  $v \in Adj[u]$  do
            ABR-Relax( $u, v, d_{uv}, b_{uv}, pd_{uv}$ );
    else
        ABR-Initialize( $G,s$ );

```

The subroutines ABR-Initialize(G,s), ABR-Extract-Min(Q,u,d) and ABR-Relax($u, v, d_{uv}, b_{uv}, pd_{uv}$) are described as follows:

ABR-Initialize(G,s)

```

For each node  $v \in V[G]$ 
    Do  $d_v \leftarrow \infty$ ;  $\pi_v = NIL$ ;
     $d_s \leftarrow 0$ ;  $pd_{ij} \leftarrow \infty$ , for  $c_{ij} < c_{st}$ ;
     $b_{ij} = \min[b_{ij}, b_{ji}]$ ;  $b_{min}^i = \infty$ ;  $bl^i = NIL$ ;

```

ABR-Extract-Min(Q,u,d)

```

 $d_{min} \leftarrow \infty$ ;  $u \leftarrow NAN$ ;
For  $i \in Q$  do
    If ( $d_i < d_{min}$ )
        {  $d_{min} \leftarrow d_i$ ; }
 $u \leftarrow i$ ;
If ( $b_{min}^u > Kc_{st} * d_u$ )
    RETURN TRUE;
else
     $d_{*\pi(bl^u),bl^u} \leftarrow \infty$ ;
    RETURN FALSE;

```

ABR-Relax($u, v, d_{uv}, b_{uv}, pd_{uv}$)

```

If ( $d_v > d_u + d_{uv}$ )
    If ( $b_{min}^u < b_{uv}$ )
        {  $b_{min}^v = b_{min}^u$ ;  $bl^v = u$ ; }
    else
        {  $b_{min}^v = b_{uv}$ ;  $bl^v = v$ ; }
     $d_v \leftarrow d_u + d_{uv}$ ;
     $b_v \leftarrow b_u + b_{uv}$ ;
     $\pi_v \leftarrow u$ ;

```

VBR routing algorithm

VBRSW-DIJKSTRA $G(V,E)$

```

VBRSW-Initialize( $G,s$ );
 $S \leftarrow \{\}$ ,  $Q \leftarrow V[G]$ ;
while  $Q \neq \emptyset$  do
    If (VBRSW-Extract-Min( $Q, u, d$ ) = TRUE)
         $S \leftarrow S \cup \{u\}$ ;
        If  $u = t$  found;
        For each node  $v \in Adj[u]$  do
            VBRSW-Relax( $u, v, d_{uv}, c_{uv}$ );
    else
        VBRSW-Initialize( $G,s$ );

```

The subroutines VBRSW-Initialize(G,s), VBRSW-Extract-Min(Q,u,d) and VBRSW-Relax(u, v, d_{uv}, c_{uv}) are described as follows:

VBRSW-Initialize(G,s)

```

For each node  $v \in V[G]$ 
    Do  $d_v \leftarrow \infty$ ;  $\pi_v = NIL$ ;
     $d_s \leftarrow 0$ ;  $d_{ij} \leftarrow \infty$ , for  $c_{ij} < \rho$  or  $b_{ij} < \sigma$ ; otherwise  $d_{ij} = 1$ ;
     $c_{min}^t = \infty$ ;  $bl^t = NIL$ ;

```

VBRSW-Extract-Min(Q,u,d)

```

 $d_{min} \leftarrow \infty$ ;  $u \leftarrow NAN$ ;
For  $i \in Q$  do
    If ( $d_i < d_{min}$ )
        {  $d_{min} \leftarrow d_i$ ; }
 $u \leftarrow i$ ;
If ( $c_{min}^u > (\sigma * d_u) / (W_{st} + \sigma * d_u / c)$ )
    RETURN TRUE;
else
     $d_{*\pi(bl^u), bl^u} \leftarrow \infty$ ;
    RETURN FALSE;

```

VBRSW-Relax(u, v, d_{uv}, c_{uv})

```

If ( $d_v > d_u + d_{uv}$ )
    If ( $c_{min}^u < c_{uv}$ )
        {  $c_{min}^u = c_{min}^u$ ;  $bl^v = u$ ; }
    else
        {  $c_{min}^v = c_{uv}$ ;  $bl^v = v$ ; }
 $d_v \leftarrow d_u + d_{uv}$ ;
 $\pi_v \leftarrow u$ ;

```

8 BIOGRAPHY

Dirceu Cavendish received his B. S. degree in electronics from Federal University of Pernambuco, Brazil, in 1986. From 1986 to 1991 he worked for telecommunication division of Philips, Brazil. From 1991 to 1994 he was an exchange student, supported by Mombusho, the Ministry of Education, in Japan. He received his M. S. degree in Computer Science from Kyushu Institute of Technology in 1994. Since then he has joined the Computer Science Department of the University of California, Los Angeles, where he is currently a graduate student of the computer networks Ph.D. program. His research interests includes congestion control and routing in high speed networks.

Mario Gerla was born in Milan, Italy. He received a graduate degree in engineering from the Politecnico di Milano, in 1966, and the M.S. and Ph.D. degrees in engineering from UCLA in 1970 and 1973, respectively. He joined the Faculty of the UCLA Computer Science Department in 1977. His research interests cover the performance evaluation, design and control of distributed computer communication systems and high speed computer networks (B-ISDN and Optical Networks).

Mobiware: QOS-aware middleware for mobile multimedia communications

Andrew T. Campbell

Department of Electrical Engineering and

Center for Telecommunications Research

Columbia University, New York, NY 10027-6699

<http://comet.ctr.columbia.edu/~campbell>

<http://comet.ctr.columbia.edu/wireless>

Abstract

Next generation wireless communications system will be required to support the seamless delivery of voice, video and data with high quality. Delivering hard *Quality of Service (QOS)* guarantees in the wireless domain is complex due to large-scale mobility requirements, limited radio resources and fluctuating network conditions. To address this challenge we are developing a QOS-aware middleware platform called *mobiware* which contains the complexity of supporting multimedia applications operating over wireline and wireless ATM networks. Mobiware is a software middleware platform based on xbind and CORBA technology that is designed to operate between the application and radio-link layers of future wireless media systems. Mobiware provides value-added QOS support by allowing mobile multimedia applications to operate transparently during handoff and periods of persistent QOS fluctuation.

Keywords

Middleware, mobile communications, adaptive algorithms, active transport, QOS

1 INTRODUCTION

Recent years have witnessed a tremendous growth in the use of wireless communications in business, consumer and military applications. The number of wireless services and subscribers has expanded with systems for mobile analog and digital cellular telephony, radio paging, and cordless telephony becoming widespread. Next generation wireless networks such as *wireless ATM (WATM)* will provide enhanced communication services such as high resolution digital video and full multimedia communications.

The main challenge in a combined wireline/wireless ATM network derives from *complexity*. Complexity is present in various forms in mobile multimedia communications. First, the combination of multi-rate multimedia connections with mobility proves difficult to achieve in practice. A connection with certain capacity reserved at a particular cell may have to be rerouted when the mobile device changes its location. The new path to the desired location may not have the original required capacity. Therefore, renegotiation of resources allocated to the connection is needed. At the same time, the flow (e.g., audio or video) should be transported and presented ‘seamlessly’ to the destination device with a smooth change of perceptual quality. This motivates the need for QOS with mobility.

Next, audio and video flows are characterized by the production, transmission and consumption of single media streams (both unicast or multicast) with associated QOS. For multicast flows, individual receivers (both wired and wireless) may have differing capability to consume flows [11]. This could be due to either fluctuating network resources with mobility or imposed by individual applications. Bridging this heterogeneity gap [14] in mobile multicast environments [12] while simultaneously meeting the individual mobile devices’ QOS requirements is an area of research that remains to be resolved.

Third, radio channel’s varying QOS characteristics and device mobility, fundamentally impact our ability to deliver hard QOS guarantees in the WATM environment. QOS controlled mobility and a QOS adaptive transport system share a common link in that they must be able to respond or “adapt” to the changes in the delivered quality due to QOS-fluctuating wireless channel or application-level mobility, respectively.

In this paper we propose a unique solution to the overall problem of complexity that is based on a methodology of networking programming based on a QOS-aware middleware platform called *mobiware*. We extend an open control methodology that has been previously developed at Columbia [7] to control ATM networks to the mobile ATM domain. The basic tenet of this methodology is to separate the network hardware from software.

The structure of this paper is as follows. Section 2 presents an overview of the

mobiware and its architecture which provides a framework for network programming and adaptation of flows. Following this we describe a set of adaptive and active algorithms which lay at the heart of mobiware's adaptation strategy. We describe a QOS controlled handoff algorithm in section 3, an adaptive and active transport algorithm in section 4 and an adaptive network service in section 5. Finally, in section 6 we provide some concluding remarks.

2 MOBIWARE

Mobiware is a software middleware platform that runs seamlessly on mobile devices, base stations and mobile-capable ATM switches. The platform is built on distributed system and Java technology and incorporates new architecture and novel adaptive algorithms to support QOS controlled mobility. The goal of the mobiware adaptive algorithms is to transport scalable flows, reduce handoff dropping and improve wireless resource utilization. We use the term "controlled QOS" in this paper to distinguish it from hard QOS guarantees offered by fixed ATM networks. Implicit in the term is the notion that flows can be represented and transported as multi-layer scalable flows at the mobile device. Adaptive algorithms help scale flows during handoff based on the available bandwidth and an application-specific *flow adaptation policy* [13]. This policy characterizes each audio and video flow as having a minimum QOS layer and a number of enhancements.

Mobiware provides a highly programmable platform for ease in the service creation, monitoring and adaptation of multimedia flows during handoff. The concepts of programmability and adaptability are fundamental when addressing the complexity of supporting mobile multimedia applications over QOS-varying mobile networks. By adaptability we mean as mobiles roam mobiware's adaptive algorithms conspire to scale flows to match available bandwidth at the bottleneck node, e.g., the base station. By programmability, we mean that mobiware's APIs are 'high-level' enough to allow adaptive algorithms to be implemented using distributed systems technology.

2.1 Architecture

Mobiware promotes the separation between mobile signalling and adaptation management on the one hand and media transport on the other. As illustrated in Figure 1 mobiware utilizes xbind (based on CORBA) and Java for signalling and adaptation management during handoff or periods of persistent QOS fluctuation. The Java Virtual Machine executes on mobile devices, base stations and mobile-capable ATM switches and supports the dynamic execution of *active transport objects (ATOs)*. These transport objects constitute an 'active' component of the mobiware transport system which can

dispatch ATOs to strategic points in the network or end-systems to provide value-added QoS support. The concept of ATOs is derived from work on protocol boosters [18].

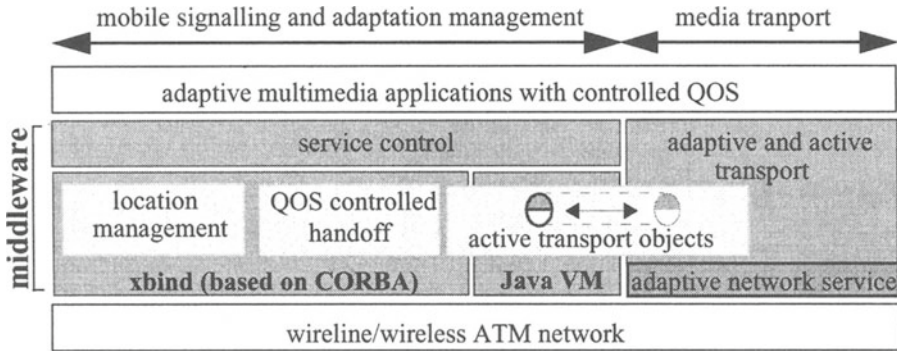


Figure 1: Mobiware Architecture

The realization of end-to-end QoS control and the exploitation of scalable flows is achieved in mobiware through; i) resource binding between mobile devices, base stations and ATM switches; and ii) provision of a set of QoS-aware adaptive algorithms. These algorithms operate in unison under the control of mobiware:

- *QOS controlled handoff*, provides signalling for handoff which exploits the use of: i) soft-state and hard-state to represent flows; ii) aggregation of flows to/from mobile devices; and iii) routing and QoS renegotiation anchor points to limit the impact of small-scale mobility on the wider fixed network;
- *adaptive network service*, provides hard QoS guarantees to base layers (BL) and soft QoS guarantees to enhancement layers (viz. E1 and E2) of multimedia flows based on the availability of resources in the wireless environment; and
- *adaptive and active transport*, supports the transfer of multi-layer flows through the provision of a QOS-based API and a set ATOs (e.g., media scaling [14]) and *static transport objects (STOs)*, e.g., playout control. STOs are statically configured and execute at mobile and fixed devices only. In contrast, ATOs are dynamically dispatched to the mobile devices, base stations or ATM switches to support value-added QoS at strategic nodes.

Media scaling plays an important role in processing flows on-the-fly to match available resources at a bottleneck node, e.g., the air-interface. Media scaling exploits the intrinsic scalable properties of multi-resolution audio and video flows. In addition, adaptive algorithms take into account the knowledge of the user's flow adaptation

policy to actively *filter* flows at critical nodes. This is achieved using an adaptive and active transport system to dispatch media scaling agents called *mobile filters* at critical nodes in the network or end-systems. Mobile filters are one of a class of ATOs which best utilize the available bandwidth to seamlessly deliver media with smooth change in the perceptual quality during handoff. Mobiware updates a location management algorithm during handoff. The mobiware location management uses a logical-name-to-logical-name mapping to express the current position of each mobile device

The mobiware platform models the wireless portion of the ATM network as being divided into pico-cells each served by a base station connected to a wired ATM network as illustrated in Figure 2. Base stations are cell relays which translate the ATM cell headers from radio ATM format to that used by standard ATM. Each base station supports signalling, QOS control and adaptation of flows based on semantics of an adaptive network service. The existing wired ATM network provides connectivity between base stations. We organize the wireless network into domains. A domain consists of a set of base stations which are under the management of mobile-aware ATM switches. A domain corresponds to a logical partition of the wireline network and the physical location of the base stations in hierarchies for scalable routing (in ATM Forum PNNI routing, these domains are peer groups).

3 QOS CONTROLLED HANDOFF

The goal of the QOS controlled handoff algorithm is to dynamically reroute a set of connections associated with a mobile device from one base station to another without significantly interrupting flows in progress. The design of QOS controlled handoff is driven by two conflicting design goals: i) support the mobility; and ii) minimise the impact that small scale mobility has on the wireline portion of the network during handoff. To achieve these two design goals we introduce:

- *mobile soft-state*, models the dynamics of mobility through the continuous re-routing and QOS re-negotiation of flows as a mobile device roams. Soft-state is established between a per mobile *QOS re-negotiation anchor point (QRP)* and the mobile device. Hard-state is used between the QRP and the fixed network portion of the network;
- *connection group (CG)*, provides a common routing representation for all virtual paths and virtual circuits destined to/from the same mobile. Connection groups decouple handoff re-routing from resource allocation at a per mobile *routing anchor point (RAP)*. The RAP allows collective control and mobility management of all flows associated with a mobile device during handoff. Having a single reference point to manage all connections greatly simplifies the hand-off; and

- *logical anchor points*, provide an interface between the hard-state and soft-state portions of flows. The RAP and QRP are logical anchor points which localize the periodic rerouting and renegotiation and during handoff processing, respectively.

Mobiware allows collective control and management of all connections associated with a mobile device using a single connection group identifier (CGI), which uniquely represents a single reference point to manage all connections. Connection groups are setup using multicast connection management operations, e.g., *addBranch* to a connection group tree. Removing a branch of a connection group tree after handoff is managed automatically through the semantics of the soft-state operations used in the mobile environment.

Mobiware handoff achieve this through the interaction of distributed QOS handoff algorithms with a set of mobiware *virtual resource objects* which model physical hardware devices and QOS as CORBA objects [8]. Mobiware models base stations [12] as a set of virtual resource objects: *virtualBasestation*, which is used to represent and manipulate the GCI/VP/VC routing table; *virtualWirelessLink*, which is used to represent and allocate QOS to a flow based on the concept of a *scheduler region* [7] [6]; and *virtualQOSFilter*, which is used to scale media at the base-to-mobile link.

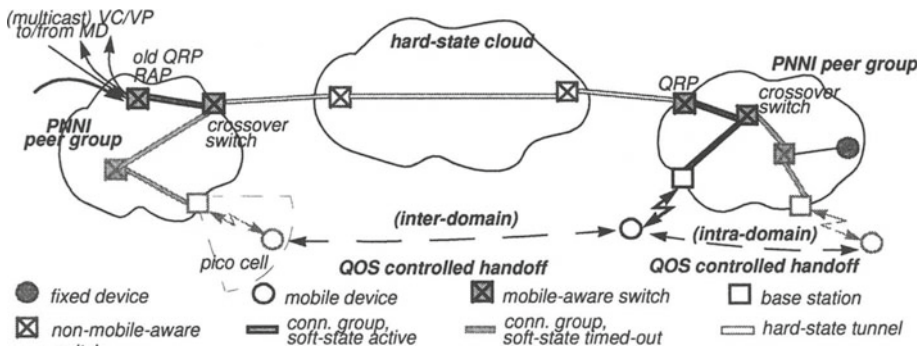


Figure 2: QOS Controlled Handoff

In the following sections we will step through the QOS controlled handoff. See Figure 3 for illustration of the handoff phases.

3.1 Signalling Phase

The first phase of QOS controlled handoff (see Figure 3 (b)) determines whether a new base station can provide a stronger signal at the desired level of QOS. A QOS monitoring algorithm resident at mobile devices monitors beacon messages which are periodically broadcast by all neighbouring base station. In addition to indicating the strength of neighbouring base station signals beacon messages indicate the residual capacity currently

available at base stations. By periodically monitoring beacons from all neighbouring base stations the mobile device is able to determine link qualities and occupancy of adjacent base stations and use this QOS state information as a basis to initiate handoff after a suitable dwell time.

The next phase of handoff is the establishment of a new signalling channel between the mobile device and the new base station. A mobile device issues a (1) *signalRequest* to the new base station over a dedicated meta signalling channel using the base station address found in the beacon message. This results in the creation of the signalling channel when the base station responds with a (2) *signalResponse*. The signalling channel carries all signalling and QOS management messages between the mobile and wireline network.

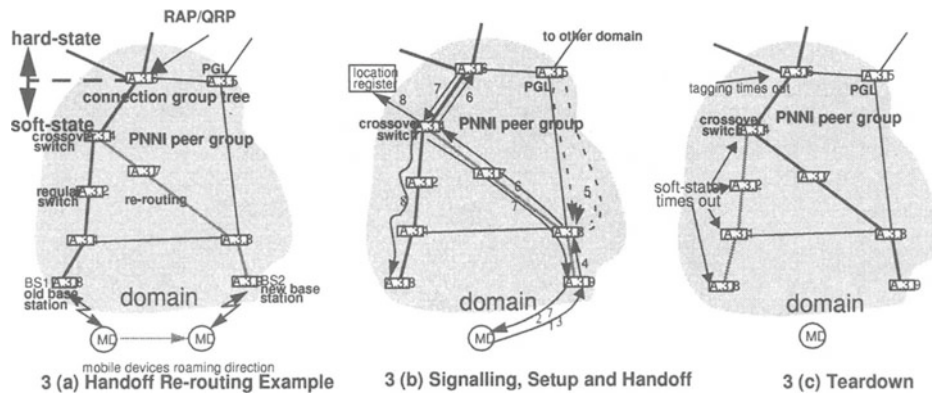


Figure 3: QOS Controlled Handoff Walk-through

3.2 Setup Phase

Once the signalling channel has been successfully created the mobile device initiates a forward handoff to the new base station. It does so by issuing a (3) *reservation message* (for details on the *res* message see section 5) which includes connection group route state information and desired QOS required. The handoff management algorithm located at the new base station uses this state information to establish a new branch (between the crossover switch and new base station) to the existing connection group tree with the desired QOS. Mobile devices express desired QOS in terms of the semantics of the adaptive service and connection groups. Connection group QOS requirements are specified in terms of connection group base layer requirements and enhancement layer requirements, respectively. Admission control located at the new base station first determines whether sufficient resources are available to support the requested handoff.

Reservation messages can be updated by the distributed handoff algorithm as they are forwarded from the new base stations toward the rendezvous switch based on the availability of resources at the traversed nodes. The semantics of the adaptive service provides hard guarantees to the base layers and admits enhancements layers based on the availability of residual resources; that is, the resources remaining once all base layers have been guaranteed. The new base station only drops the handoff (*handoffDrop*) if insufficient residual capacity is available to meet the group connection base layer resource requirements. Assuming that sufficient resources are available to meet the minimum QOS requirements, the *res* message is routed toward the crossover and rendezvous switch reserving resources on route. Connection group routing information is used in combination with PNNI routing to determine (5) the shortest path between the new base station and the existing connection group tree which meets the desired QOS reflected in the *res* message. The PNNI peer group leader is interrogated (5) should the mobile device roam outside the current peer group domain of the old base station. This may suggest that the existing RAP would provide a sub-optimal routing point and a new RAP along with a new QRP. Generally the new QRP is located close to the mobile device and over a longer timescale a RAP rerouting algorithm determines when and where to move the RAP.

In the case where roaming is within the current peer group domain, the new base station issues a (6) *res* message to the crossover point. Each intermediate switch on route between the base station, crossover point and QRP provides admission testing and resource reservation. The QRP terminates QOS renegotiation. The existing connection group traverses the crossover switch. Generally, the crossover switch processes the *res* message and forwards it to the rendezvous switch which also processes it and then responds by sending an (7) *adaptation message* (for details on the *adapt* message see section 5) to the mobile via crossover and the new base station. The *adapt* message is used to commit switch and base station resources at the downstream nodes for the new connection group branch. Once the base station receives the *adapt* message it commits resources and broadcasts the *adapt* message to all mobile devices in the cell.

3.3 Handoff Control

The *adapt* message serves two purposes. First, it confirms the level of QOS provided by the new base station and wireline portion of the network to the new mobile device as it enters a new cell. Second, it informs existing mobile devices of any resource changes which may have occurred during handoff to accommodate the new mobile device. The semantics of the adaptive service states that base layer resource reservation requests receive precedence over requests for any higher layer qualities. To this extent the quality delivered above the base layer to existing mobiles may be altered to allow a new mobile device to enter a new cell.

The handoff algorithm interacts with the media scaling agents (which interact with mobile filters - see section 4.2) at the base station and QRP during connection group setup. The level of media scaling is dependent on the current utilization of the wireless link, application specific desired QOS and the semantics of the adaptive service. Media scaling may result in the “scaling-down” of mobile delivered quality when a mobile device enters a pico-cell and “scaling-up” when they leave. Mobile devices up-load Java-based mobile filters (see section 4.2 on active transport objects) to the base station or rendezvous switches in the case where the requested QOS could not be supported. This results in enhancement layers being dropped or filtered at specific points on a connection group tree (e.g., at the new base station or QRP). QOS filters support the delivery of different combinations of layers to particular mobile devices based on the available resources. Network-based mobile filters are essential to support multicast QOS to heterogeneous receivers.

Handoff registers new mobile devices with the domain location management which in turn allocates a new proxy ATM address as mobile devices roam into cells within a new peer group domain. A (8) *locationUpdate* message is used to register the mobile devices at the home location in this case and update the cache register at the old base station (8). QOS controlled handoff is based the notion of soft-handoff for roaming mobile receivers and hard-handoff for roaming mobile senders. In the case of soft-handoff, the mobile device simultaneously interacts with the old and new base stations. Once the QRP responds with an adapt message to the mobile device it begins to forward the new flow to the mobile. This results in duplicate cells (for old and new flow) arriving at the mobile device via the old and new base stations. The rendezvous switch uses cell tagging to preserve ATM cell level sequence integrity at the mobile device during handoff.

After tagging has commenced mobile devices determine suitable synchronization points between old and new flows and initiate flow switching from the old to the new flow. After flow switching the old flow is rendered redundant. Old flows continue to arrive at the mobile devices as long as the route between the old base station and QRP is active; that is, old flows are switched through to the mobile while the mobile soft-state is still installed and has not timed-out. Mobile devices do not, however, have to process old flows.

3.4 Teardown

After flow switching the new base station refrains from sending any further periodic *res* messages to the old base station and closes the signalling channel to the old base station. Once the mobile soft-state timer expires the old branch of the connection group tree between the QRP to the old base station is timed out and removed; that is, after the mobile soft-state timer expires resources are deallocated and switching tables flushed accordingly. This is defined as teardown.

Media scaling is once again invoked at the old base station to determine if deallocated resources can be utilized by the any existing mobile devices at the old base station. Mobile devices located at the old and new base station periodically probe the base station and network for more resources using the *res* and *adapt* message pairs. During handoff, mobile devices resident at the old base station issue reservation messages toward the QRP and receive an *adapt* message which indicates any extra resources made available after a mobile has left the current cell. The new *adapt* message reflects any new resource availability and adjusts any QOS filters via interaction with media scaling agents at the old base/ rendezvous switch. We describe the condition whereby resources freed up during handoff are distributed to remaining mobile devices at the old base station as scaling-up.

4 ADAPTIVE AND ACTIVE TRANSPORT SYSTEM

A fundamental aspect of our work is the development of an adaptive and active transport that incorporates a QOS-based API and a full range of transport algorithms to support the delivery of continuous media over mobile networks. The mobiware transport operates in two modes:

- *adaptive mode*, which provides a set of STOs (viz. playout control, flow control, flow scheduling and shaping, flow monitor and adaptation manager) that best assists multimedia applications when adapting to minor QOS fluctuations as a consequence of cell/packet loss and delay variation; and
- *active mode*, which provides a set of ATOs (viz. mobile filters [17], mobile error control [19] modules and mobile snoop [15] modules) that can be dynamically dispatched to mobile devices, base stations or mobile-capable ATM switches to provide value-added QOS during conditions of persistent QOS fluctuation that may emerge during handoff.

In the active mode, local adaptation manager monitor the loss available bandwidth characteristics of flows and interact with ATO control to select, dispatch, bootstrap, configure and tune the appropriate ATO to the requesting target node.

4.1 Transport API

The transport API formalizes the end-to-end QOS requirements of the adaptive multimedia application (client or server) and the potential degree of media scaling acceptable to the application [13] (based on flow adaptation policy) which bounds its perceptual range. As illustrated in figures 1 and 4, applications interact directly with the service control API to establish, control and maintain the requested service. The service

control algorithm governs the point at which a QOS controlled handoff is initiated.

The adaptive transport API assumes a client-server model where servers interact with service control to create *QOS groups* specifying their *QOS profile* (i.e., QOS requirements: traffic class, delay and bandwidth) for each multi-resolution of the flow and *flow adaptation policy* (i.e., the type of coding and prioritizing of the various resolutions used, and flow-spec for each flow) of the source media. The traffic class and delay bounds are common for each resolution of the scalable flow. The user can prioritize connections so that during handoff certain connections receive preferential treatment over others in light of reduced bandwidth (e.g., drop the video connection before the audio). The bandwidth for each resolution is specified in a flow-spec [13] by the clients and servers. Clients join QOS groups, inspect the QOS profile of the source and then select the appropriate resolutions by matching their capability to consume source media. For full details on the adaptive algorithms and API see [13].

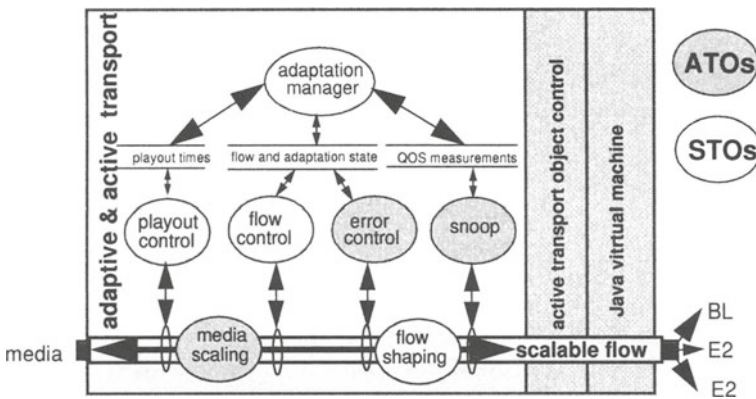


Figure 4: Adaptive and Active Transport System

4.2 Active Transport Objects

Adaptation managers operate on all mobile devices, base stations and mobile-capable ATM switches and continuously monitor the performance of the channel and each flow. Adaptation managers and QOS controlled handoff algorithms interact with local ATO control to request the remote loading of ATO to support QOS during periods of service degradation.

Currently mobiware supports three types of ATOs:

- *mobile filters*, which are used during periods of limited bandwidth to either drop layers of a flow (we call these types of mobile filters media selectors) or process audio and video in the compressed domain to meet an available bandwidth [17]

[11] [9]. In general mobile filters are dispatched once during handoff and are tuned based on the available bandwidth.

- *mobile error control modules*, which provide hybrid ARQ and FEC mechanisms for improved reliability of audio, video and data over the air-interface during periods of excessive cell/frame loss [19] and
- *mobile snooping modules*, which help increase the performance of flows (e.g., TCP data flows) by snooping [15] [19] end-to-end protocol messages as a means to trigger the local value-added error control mechanism over the air-interface.

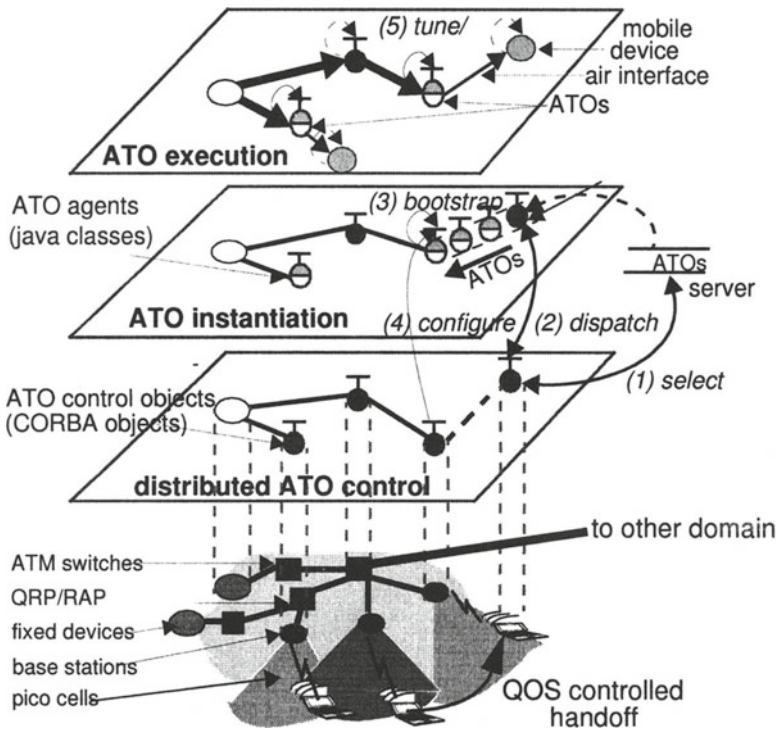


Figure 5: Active Transport Object Management

ATO are application specific and interact with differing algorithms to provide value added support. For example mobile filters are driven by an available bandwidth indication and interact explicitly with the adaptive network service algorithm. In addition, “*ATO state*” can be transparently moved during handoff. For example, a mobile filter executing on a base station can be automatically moved and reconfigured at the new base station. Movement of ATO state like this is dependent of the specific ATO and the

operating conditions existing in the new cell. ATO state can propagate along with the connection state from the old to the new node over the wireline portion of the network. In this case the ATO state can take advantage of the wireline ATM interconnect to achieve a fast handoff. ATOs are capable of flexible autonomous actions in QOS fluctuating environments. Operationally, ATOs can be dispatched, configured and executed at any ATO-capable networked node.

4.3 Active Transport Object Management

ATO management consists of a distributed algorithm which manages the installation of new ATOs anywhere in the network. ATO management uses a client-server approach to select, dispatch, bootstrap, configure and tune new ATOs between an ATO server and the target node.

As illustrated in Figure 5 ATO management is divided into three operational modes:

- *ATO control* is a distributed signalling algorithm which comprises mobiware ATO control objects. These objects are permanently resident at the base stations, mobile capable ATM switches and mobile devices. ATO control objects support a set of methods to *select* (1), *dispatch* (2) and *configure* (4) mobile filters;
- *ATO instantiation* fetches remote Java bytecode classes (which are representations of ATOs) and *bootstraps* (3) them into Java VM environment based at base stations or mobile capable ATM switches. Once an ATO has been loaded and booted into a switch or base station the local ATO control object initiates a *configure* (4) operation to complete the instantiation phase; and
- *ATO execution and tuning*, at this point ATO can act autonomously to provide value-added QOS support to the transport of continuous media or data. Some ATOs interact with existing algorithms during the execution phase and can be periodically tuned (5).

As illustrated in figure 4, the adaptation manager periodically registers the monitor the QOS delivered at the channel. In the case where the platform needs value added software. Mobile filters are capable of flexible autonomous actions in QOS fluctuating environments. Operationally, mobile filters can be dispatched, configured and executed at the base stations or ATM switches. Mobile filters can be periodically tuned via a filter interface to match the available resources at a particular bottleneck node (e.g., base station or mobile capable ATM switches). In addition to being QOS adaptive, mobile filters automatically propagate during handoff. For example, in the case of a handoff between two WATM radio ports existing mobile filters (called *mobile filter state* in mobiware) propagate along with the connection state from the old to the new base station over the wireline portion of the network. In this case the mobile filter state can take advantage of the wireline ATM interconnect to achieve a fast handoff.

4.4 Mobile Filter ATOs

Currently we have implemented a media selector ATO in Java that drops either E1 (i.e., P pictures) and E2 (i.e., B pictures) frames of an MPEG-1 stream based on the available resources. Media selectors ATOs do not process the media unlike other computationally intensive mobile filters e.g., dynamic rate shaping mobile filters [9]. One of the key performance issues related to mobile filter technology is the time taken to dispatch, bootstrap and configure new agent over wireless and wireline interfaces. Another important performance concern relates to the performance penalty paid by flows as they are processed at ATM switches and base stations. The amount of delay introduced by such operations as flows traverse ATO is dependent on the computational complexity of the ATO, the additional overhead of the Java VM and the cost of derailing ATM cells for post filtering. For some initial performance results relating mobile filter ATO see [17] [19].

5 ADAPTIVE NETWORK SERVICE

The adaptive service is based on previous work on adaptive service for wireline ATM networks first proposed in [13]. This service model provided “hard” guarantees to the base layer (BL) of a multi-layer flow and “fair share” guarantees to each of the enhancement layers (e.g., E1 and E2) supported by the service. To achieve this, the BL undergoes a full end-to-end admission control test [13]. In contrast, enhancement layers were admitted without any such test but competed for residual bandwidth with all other adaptive flows which traversed a particular switch along a specific route.

Similar schemes have subsequently been adopted by a number of practitioners in the wireless communications field [4] [5]. However, one draw back of these adaptive service schemes is that a QOS fluctuation at a particular switch (e.g., due to a new call being admitted) in a wireline/wireless environment can potentially impact other flows traversing the link/port. This results in a chain-reaction as distributed adaptive resource management algorithms resolve the new change in state. Unbounded adaptive service will typically occur in highly QOS fluctuating environments such as in mobile wireless systems. Therefore, there is a need to bound this chain-reaction while at the same time offering an adaptive service to mobile terminals to reduce the probability of handoffs being dropped as new mobiles roam into a bottleneck pico-cell.

This is the motivation behind our adaptive service. The adaptive service primarily operates over the wireless segments of end-to-end flows; that is, between the QRP, base station and the mobile devices as illustrated in Figure 2. Therefore, QOS adaptation is limited to the pico-cellular area and does not directly impact the wired portion of the network. Mobile devices compete for wireless resources by interacting with an adaptive

service algorithm at the base station. The adaptive service provides “hard” guarantees to the BL and “fair share” guarantees to each of the enhancement layers (E1 and E2). This limits adaptation to where it is more likely to occur: at the wireless link between the base station and QRP. Enhancement layers are rate controlled based on explicit feed back mechanisms about the current state of the on-going flow and the availability of residual bandwidth at a base station.

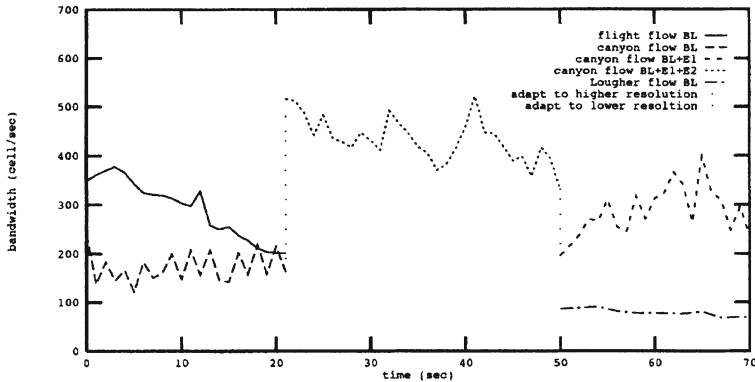


Figure 6: Scalable Video Flows

A number of objectives motivate the design of a new wireless adaptive service for mobile QOS fluctuating networks. The first objective is to admit as many base-layers as possible across the wireless link. As more base-layers are admitted the guaranteed capacity region grows to meet the hard guarantees for all base signals. In contrast, the residual capacity region shrinks as enhancement layers compete for diminishing residual bandwidth resources. Our second objective is to fairly share this wireless residual capacity among competing enhancement layers based on an algorithm called *weighted fairshare* [13]. In addition, another associated objective is to limit the impact of the wireless adaptive service on the wired network. Our fourth and final objective is to adapt flows both discretely and continuously based on an adaptation mode supplied in the user-supplied adaptation policy. In the discrete mode no residual bandwidth is allocated by the wireless adaptive service algorithm unless a complete enhancement can be accommodated. In contrast, in continuous mode any portion of the residual capacity can be made available and be utilized by the adaptive flow [12].

Figure 6 illustrates the operation of the adaptive service over a wireline ATM link. The bandwidth of the link which the adaptive algorithm shares between three different video flows (akin to mobile devices) was setup to be similar in bandwidth capability to a low bandwidth wireless ATM link (e.g., admission control is preset to 600 ATM cells/sec to be shared by all competing flows). The scenario shows the consumption of three video clips beginning at time zero. The *canyon* and *flights* video flows start at time zero and

only have their BL supported; that is the minimum quality is provided. No residual capacity is available to support higher qualities because of the limitation of the available resource (i.e., 600 cells/sec). Twenty seconds into the trace the *flights* video flow terminates freeing up resources for remaining flows (i.e., remaining mobile devices). This is akin to a mobile device roaming out of a pico-cell. At this point the adaptive service attempts to switch into the higher resolutions for remaining flows. The trace shows that the *canyon* video flow receives the best quality (i.e., BL+E1+E2) after the *flights* video terminates. This situation remains stable until another video flow starts up at 50 seconds into the trace which is akin to a mobile device roaming into the cell and competing for wireless link resources. Resources are thus allocated to meet the BL QOS requirements of the new mobile device. The *canyon* video flow is adapted down to the BL+E1 quality at 50 seconds into the scenario as a result of admitting the new flow (akin to a new mobile). While the scenario is taken from wireline ATM experimentation into adaptive services [13] the results indicate what the service would look like to mobile devices as they roamed between cells.

5.1 Mobile Soft-State

Rendezvous switches serve as points above which re-routing and QOS re-negotiation only occur when a mobile device roams between two adjacent routing domains. Inter-domain roaming operates at a much lower frequency than intra-domain roaming. Occasional inter-domain roaming requires re-routing and QOS re-negotiation in the wireline network. We contain the frequency of re-routing and QOS re-negotiation caused by small-scale mobility from impacting the wider wireline network. This is contained at the QRP. Above the QRP only infrequent re-routing and QOS re-negotiation occur. Below the QRP (between the mobile devices and QRPs) frequent re-routing and QOS may be observed.

Based on our understanding of the dynamics of small-scale (intra-domain roaming) and large-scale (inter-domain roaming) mobility we model end-to-end flow through a combination of “hard-state” (above the QRP in the wireline network) and “soft-state” (below the QRP in the wireless network). We argue that soft-state is suited to support the dynamics of mobility and QOS adaptation found in wireless and mobile networking.

A significant contribution to our approach to handoff is the use of soft-state [16] to support the dynamics of mobility in the wireless and wireline network. Soft-state is used between the mobile devices and QRP because it best suits the dynamic nature of QOS adaptation and device mobility. Mobile devices periodically send reservation messages (*res*) toward the QRP in the wireline network. These reservation messages carry the mobile devices desired QOS requirement and are interpreted by the base station and fixed network infrastructure. Resources are allocated to the mobile device over a particular wireless/wireline route for the duration of the mobile soft-state timeout. If during that

time the infrastructure receives another *res* message it refreshes the state held in the base station and switches between the mobile device and the QRP. As mobile devices roam between adjacent cells the periodic *res* messages are used by the mobile devices to establish a new state (i.e., a route with QOS attached) at the new base station and intermediate switches. In addition, as the mobile device roams into a new cell the old connection group soft-state times out and is deallocated automatically. The periodic *res* message is used in combination with an adapt message to continually probe the base station and network for better QOS.

6 CONCLUSION

This paper has introduced a QOS-aware middleware platform for mobile multimedia communications. Mobiware has been specifically designed to address the complexity of proving QOS support for adaptive multimedia applications over wireless and mobile networks. Mobiware includes two key attributes which contains complexity. These are programmability and adaptability. In this paper we presented the key adaptive algorithms that govern the available strategies for adaptability in mobiware. These include a QOS controlled handoff scheme which promotes the use soft-state, connection groups and logical anchor points for fast and seamless handoff. We have also described a new transport systems which utilizes adaptive and active transport objects to provide value added QOS support in the end-systems and network. The final mobiware adaptive strategy is based of a novel adaptive network service which has been designed to provide hard QOS guarantees for minimum flow quality and best effort delivery for higher quality.

The mobiware testbed consists of 4 ATM switches (viz. ATML Virata, Fore ASX100/ASX200s, NEC Model 5, Scorpio Stinger) and 4 base stations. The base stations are multihomed 200 MHz Pentium with 25 Mbps wireline access to the wireline ATM network and 2 Mbps WaveLan air-interfaces to a number of mobile devices based on Pentium PCs and notebooks. The PCs run Linux, Windows/NT and xbind (based on CORBA). An early version of mobiware runs on PCs, base stations and the ASX100 ATM switch

Finally, we have implemented a beta version of the handoff protocol which support soft-state, connection groups and logical anchor points [19]. In addition we have completed the implementation of active transport object management and support mobile filters for media selection during handoff [20]. Currently we are implementing other adaptive and active transport objects and plan to investigate the integration of mobiware with wireless ATM radios. Future work will investigate the use of more sophisticated adaptation technics to hide the interworking of different radios.

7. REFERENCES

- [1] Raychaudhuri, D., (NEC USA), Dellaverson, L., (Motorola), Umehira, M., (NTT Wireless Systems), Mikkonen, J., (Nokia Mobile Phones), Phipps, T., (Symbionics), Porter, J., (Olivetti Research), Lind, C., (Telia Research) and Suzuki, H., (NEC C&C Research), "Scope and Work Plan for Proposed Wireless ATM Working Group", ATM Forum Technical Committee, ATM Forum/96-0530/PLEN, April, 1996.
- [2] Porter, J., Hopper, A., Gilmurray, D., Mason, O., Naylor, J., and A. Jones, "The ORL Radio ATM System, Architecture and Implementation", Technical Report, ORL Ltd, Cambridge, UK, January, 1996.
- [3] Messerschmitt, D. G., Reason, J. M. and A. Y. Lao, "Asynchronous Video Coding Wireless Transport", Workshop on Mobile Computing Systems and Applications", Santa Cruz, December 8-9, 1994.
- [4] Bharghavan, V., "Adaptive Resource Management Algorithms for Mobile Computing Environment", Proc. OPEN-SIG Workshop, New York, April 16-16, 1996.
- [5] Brown, K. and Singh, S., "A Network Architecture for Mobile Computing", INFOCOM'96, San Francisco, March 1996.
- [6] Nagshineh, M., and A. Acampora, "QOS Provisioning in Micro-Cellular Networks Supporting Multimedia Traffic", INFOCOM'95, Boston, April, 1995.
- [7] Lazar, A.A., Bhonsle, S. and Lim, K.S., "A Binding Architecture for Multimedia Networks", Journal of Parallel and Distributed Computing, Vol. 30, Number 2, November 1995, pp. 204-216.
- [8] The Common Object Request Broker: Architecture and Specification, Revision 1.2, published by the Object Management Group (OMG) and X/Open, December 1993.
- [9] Eleftheriadis, A., and D. Anastassiou, "Meeting Arbitrary QOS Constraints Using Dynamic Rate Shaping of Coded Digital Video", Proceedings, 5th International Workshop on Network and Operating System Support for Digital Audio and Video, Durham, New Hampshire, April 1995, pp. 95-106
- [10] Aurrecoechea, C., Campbell, A.T. and L. Hauw, "A Survey of QOS Architectures", Multimedia Systems Journal, Special Issue on QOS Architecture, 1996, (to appear)
- [11] Nicholas Yeadon, Francisco Garcia, Andrew Campbell and David Hutchison, "QOS Adaptation and Flow Filtering in ATM Networks", Proc. 2nd International Workshop on Advanced Teleservices and High-Speed Communication Architectures, Heidelberg, Germany, September 1994.
- [12] Campbell A..T., "Towards End-to-End Programmability for QOS Controlled Mobility in ATM Networks and their Wireless Extensions", Proc. 3rd International Workshop on Mobile Multimedia Communications (MoMuC-3), Princeton, Sept. 25-27, 1996, and Wireless ATM Workshop, Espoo, Finland, Sept. 2-3, 1996, (invited presentation)
- [13] Campbell A. T. and Coulson G., "Implementation and Evaluation of the QOS-A Transport System", 5th IFIP International Workshop on Protocols for High Speed Networks, Sophia Antipolis, France, October 1996.
- [14] Delgrossi et al., "Media Scaling in a Multimedia Communications System", ACM Multimedia Systems Journal, Vol. 2., No. 4, 1994.
- [15] Balakrishnan, S. Seshan, E. Amir, R.H. Katz, "Improving TCP/IP Performance over Wireless Networks", 1st International Mobile Computing and Networking (MOBICOM'95), Berkeley, November 13-15, 1995
- [16] Zhang, L., et al., "Resource Reservation Protocol (RSVP) - Version I Functional Specification", Working Draft, draft-ietf-rsvp-spec-07.ps, 1995
- [17] Balachandran, A. and A. T. Campbell, "Mobile Filters: Delivering Scaled Media to Mobile Devices", Technical Report, Center for Telecommunications Research, Columbia University, October, 1996.
- [18] Feldmeier, D., "Protocol Boosters", COMET Group Seminar, Feb 1996.
- [19] <http://comet.ctr.columbia.edu/wireless/research/>
- [20] Campbell, A., Liao, R.-F. and Shobatake, Y., "Using Soft-State for Handoff in Wireless ATM Networks", the Sixth WINLAB Workshop on Third Generation Wireless Information Networks, March 20-21, 1997

PART FIVE

Fundamental Concepts

Enforcing quality of service for adaptive multimedia applications via fair queuing

F. Toutain

*ENST Bretagne, Département Réseaux et Services Multimédia
2, rue de la Châtaigneraie, BP 78
35512 Cesson-Sévigné Cedex, FRANCE*

tel (+33) 2 99 12 70 41

fax (+33) 2 99 12 70 30

ftoutain@rennes.enst-bretagne.fr

Adaptive applications are more and more perceived as a cornerstone of future integrated services packet networks. However, for these applications to be widely deployed, we advocate that they should be guaranteed some quality of service, in terms of floor rate, delay, and resource sharing equity. In this paper we propose an integrated approach that deals with these issues by means of a novel Generalized Processor Sharing -like scheduling scheme. Moreover we show that our proposal has the same implementation complexity as the Self Clocked Fair Queuing one, and that it provides the optimal delay bound for packet GPS-like schedulers.

Keywords: Adaptive applications, QoS, Fair queuing

1 INTRODUCTION

Adaptive multimedia applications have been more and more considered for the past few years. These are applications being able to modify their throughput in response to changing network conditions. They adapt to congestion by reducing the amount of traffic they put in the network, and conversely react to available bandwidth by increasing it, much in the same way as TCP does in the Internet. These variations obviously impact the quality of service delivered to the user, in terms of audio or video resolution, frame rate, frame size, color depth, or end-to-end delay, depending on the adaptation scheme that is used [10, 14]. Adaptive applications traditionally target so-called best-effort networks such as the Internet, where they are considered

as an alternative to deploying QoS scheduling and resource reservation schemes (e.g., in [5]). They will also be useful in future integrated services networks, not because these networks are envisioned to provide a best-effort service class, but as a means to deal with resource scarcity. Indeed, there is anecdotal evidence from computer science history that resources always end up to be scarce (RAM, peak CPU power, hard disks, ...). And despite the successful dimensioning of telephone networks, there is no evidence that integrated services networks will have so high a bandwidth and so predictable a traffic pattern that they will never be congested [3, 8, 24, 28]. This is why, for instance, service classes dedicated to adaptive applications are envisioned by the IETF for the future integrated services Internet [31]. Furthermore, it is worth noticing, as in [13], that adaptive applications may be used in a guaranteed QoS context, when it is available, and as such outperform non-adaptive ones, in terms of flexibility and usability. Hence adaptive multimedia applications design and support tend to become major trends in the multimedia networking research community. However we identify three major issues that must be tackled for adaptive applications to be successful: floor bandwidth, delay, and equity. These issues are detailed below.

The first issue is linked to common-sense deductions, and has been already described into several publications (e.g., [3, 6, 7, 12]). For short, it is: continuous media streams cannot be degraded below some threshold, or floor rate. Indeed, it is well-known that audio compression below a few hundreds bits per second produces barely understandable results. For video, different bounds, depending on the compression method as well as the application itself, could be roughly given. For instance consider that a multi-layered video stream can be easily degraded by dropping some layers, but that its baseband must not be dropped. The point here is that each media stream has a minimum bandwidth requirement, which must be fulfilled for the content to be useful. This is in contrast with traditional data communications using TCP, which may reduce to several bits per second while still reaching adequate quality of service (that is, data integrity). Furthermore, different users will have different sensitivities to the noise introduced by degrading the media content. Consider for instance people listening to a foreign language audio stream, as compared to people hearing their native language. Similarly, depending on the user itself and also on the communication purpose, the same generic video tool could place different requirements on the floor bandwidth to be guaranteed: a private user saying hello to family would probably tolerate more degradations than a professional user wishing to examine some complex diagrams. Consequently, it turns out that almost any adaptive application should be given rate guarantees for its minimum floor bandwidth needs, which should be freely selectable.

The second issue is that of end-to-end delay. A yet unknown amount of multimedia applications are targeted towards interpersonal communications (phone, video-phone, teleconferencing, teleteaching, ...). As such they require *interactive* communications, that is, end-to-end delay on the order of a hundred milliseconds

[15]. Media synchronization is even more stringent. In addition, several future applications, among which are distributed simulations, distributed games, cooperative teleworking, distributed whiteboards, ..., will be real-time applications, with very tight delay constraints [13]. Even playback applications benefit from delay bounds: (1) in order to limit their playout buffer size so as to save memory, hence production costs (set top boxes); (2) to have reasonable response times (stop, fast-forward,...). All these applications can be adaptive but they will not adapt much in the delay domain. Hence all these applications require specific guaranteed delay bounds [24].

The third issue does not address applications themselves, but rather application competition. Several studies have shown that different adaptive applications, when competing for network (or end-system) resources, do not converge to a fair equilibrium (e.g., [9, 16]). Rather, the equilibrium tends to be appealing only for the most aggressive application, since it gets the whole bandwidth while the others are doomed to service interruption. This is a major issue because the simplest way for an application to be aggressive is to not adapt to a congestion (i.e. to be greedy). Hence, unless fairness is enforced, adaptive applications are unable (and not expected to be able) to behave as “good citizens” and reach a fair equilibrium in the network sharing process, both with each other, and in the presence of non-adaptive applications. Notice that the same is true for traditional computer communications, and imposes to rely on a single congestion control algorithm (e.g. TCP in the Internet). However imposing a unique adaptation scheme for all adaptive applications is unrealistic, as it would severely affect the deployment of such applications [29]. Furthermore, even with fairness enforcement mechanisms, the issue that remains is to define fairness. The equity we will consider throughout this paper has a common-sense definition: we require every application to be affected in the same proportion by network conditions. Hence, if for instance the aggregated rates of all sessions sharing a link exceed the link capacity by 10%, then every session will have to lower its rate by 10%. This definition is indeed proportional fairness, as described in [23], where it is shown to reach optimal network efficiency provided that (1) users declare their true needs, and (2) the network allocates bandwidth in proportion to these needs. Also notice that GPS schedulers enforce proportional fairness (due to lack of space, we will not describe much the GPS approach; see [2, 11, 19, 26]).

The three issues described above have been previously treated in isolation. We will show in section 2 that these solutions cannot be combined to solve the three problems, and we will propose an integrated approach that provides floor bandwidth and delay guarantees, and enforces a fair competition between adaptive (and also non-adaptive) multimedia applications. Section 3 and 4 respectively relate to the fluid-flow (i.e. ideal) and packetized (i.e. realistic) scheduling models that we propose. Implementation is addressed by section 5, along with some delay bound results, and concluding remarks are given in section 6.

2 PREVIOUS APPROACHES

In this section, we review different proposals that were aimed at solving the floor rate, guaranteed delay, and equity issues. Since no integrated approach has been proposed yet, to our knowledge, we examine the three issues one after the other in the following subsections, and we show that they cannot be mixed into an integrated approach.

2.1 Guaranteed Delay

It is well known that a single FIFO server, shared by several sessions, provides a guaranteed bound on the queuing delay. However in this paper we envision applications having different delay constraints, and as such, requiring individual delay bounds. Furthermore, FIFO scheduling does not enforce fairness, nor does it provide rate guarantees. Individual delay bounds can be obtained by using fair queuing strategies. Depending on the buffer sizes and reserved rates, specific delay bounds can be achieved. But we will show in subsection 2.3 that using standard GPS-like queuing service raises the fairness issue.

2.2 Floor Bandwidth

Once adaptive application needs for floor bandwidth guarantees are recognized, it becomes obvious that pure best-effort networks cannot actually support these applications. This is extensively described by Lefelhocz, Lyles, Shenker and Zhang in [24], and also exemplified in [3, 28], who independently state that admission control mechanisms must be used in order to exercise congestion avoidance. Indeed, the use of guaranteed QoS schemes is advocated in these studies, along with reservation protocols which allow admission control to be done inside each node along the session path. The idea is then to make use of guaranteed service classes for the application floor rate, and to send all traffic above this minimum inside a best-effort class, where resources can be dynamically shared by several applications. Then, as the node becomes loaded, the amount of resources dedicated to the best-effort class decreases, which causes adaptive applications to lower their rates. While allowing adaptive applications to get their floor bandwidth needs, this approach has a number of drawbacks. First, it forces the applications to open two sessions inside the network: one for the guaranteed traffic, the other for the best-effort traffic. This involves additional processing to demultiplex / multiplex the original stream into end-systems. This potentially doubles the number of session contexts to be handled by the network nodes. This also requires that the two sessions be routed as a whole. Next, fairness needs to be enforced inside the best-effort service class. This imposes additional mechanisms, such as Random Early Detection (see [17]). But the main limitation is that, whereas the guaranteed part of the application traffic will have bounded delay (any value is possible, depending on the resource reservation), the best-effort part will not (unless it is a bufferless class, which is rather unrealistic). As a consequence, it may well be the case that, for some applications, the best-effort

traffic reaches the end-system too late to be used. In other words, despite fairness enforcement, the application would not get its fair share of the network.

2.3 Equity

The fairness enforced by GPS-like queuing policies is a proportional one. Here we consider using such a scheduling policy in the context of adaptive applications. A work-conserving GPS server is defined as follows. Let N sessions be characterized by positive real numbers $\phi_1, \phi_2, \dots, \phi_N$. Let $W_i(s, t)$ be the amount of session i traffic served in the time interval $[s, t]$. The inequality

$$\frac{W_i(s, t)}{W_j(s, t)} \geq \frac{\phi_i}{\phi_j} \quad j = 1, 2, \dots, N, \quad (1)$$

holding for any session i that is backlogged throughout $[s, t]$, defines the GPS server. A session is said to be backlogged if it has at least one data unit awaiting service or being serviced. It follows from (1) that, for any two sessions i, j that are backlogged throughout $[s, t]$,

$$\frac{W_i(s, t)}{\phi_i} = \frac{W_j(s, t)}{\phi_j}, \quad (2)$$

that is, the server distributes bandwidth to all backlogged sessions in proportion to their ϕ_i , which are thereby called service shares or weights.

Adaptive applications are characterized by their floor rate, delay bound requirement, and also by their maximum rate, which corresponds to the highest quality that can be reached by the application. The maximum rate is meant to depict the user's needs. Either floor rate or maximum rate can be given as the weight that is used by the GPS-like scheduler (i.e. the ϕ_i values). If the floor rate value is used, then, assuming that an admission control procedure is exercised, the floor rate requirements can be met, and, consequently, the delay bounds can also be. But then, the proportional fairness is defined over the different floor values. Based on a very simple example, we show that this kind of fairness is problematic: Consider that a link capacity $C = 10$ is shared by two adaptive applications A_1 and A_2 with floor bandwidth requirements $f_1 = f_2 = 1$, and maximum rates $m_1 = 5$ and $m_2 = 10$. Based on the floor rate proportions (i.e. 1:1), the capacity is split in two equal parts $c_1 = c_2 = 5$. Hence the first application gets its maximum rate whereas the second one must adapt to a 50% degradation. It is clear from this example that applications are not affected by congestion in the same proportion. If however we decide to give the maximum rates for the scheduling values (and hence define fairness over these values), then the allocated capacities become $c_1 = 10 / 3$ and $c_2 = 20 / 3$. It turns out that each application must adapt to a 33.33% degradation, which conforms to our idea of fairness, as detailed in the introduction. Such a fairness ensures that each application takes its part of a congestion, so that no one is forced to zero before the

others when available bandwidth is reduced. Unfortunately, the guaranteed rate for any session i is then given by:

$$x_i \geq \frac{m_i}{\sum_N m_j} \cdot C, \quad (3)$$

that is, it relates to the maximum rates of all sessions sharing the node. Hence it is not possible for an application to freely choose its floor rate, and consequently its delay bound. The trouble is that minimum requirement (that is, floor rate) and maximum requirement (maximum rate) cannot be independently stated using GPS. Another way to present this is the following: split the total bandwidth into a guaranteed part (for floor rates) and a spare part. The guaranteed part allocation is to give each session exactly its floor rate. The spare part allocation must be in proportion to users' needs. With GPS:

- Either use the floor rate to depict users' needs. In such a case, one cannot both request a low floor rate and a high need for available bandwidth. In the limiting case, a session requesting a null floor rate will get zero additional bandwidth. It turns out that such a scheme has a bias against highly adaptive sessions.
- Or use the maximum rate to depict the floor rate. Then the floor rate depends on the other sessions' maximum rates, so it cannot be individually stated. Also consider that adding admission control based on the maximum rates to gain control on the floor rates turns us back to the first case.

In the remaining of this paper, we propose a new scheduling strategy, which relies heavily on GPS, but makes it possible to independently reserve a floor rate (hence to get any desired delay bound) and still to achieve the maximum rate fairness.

3 FLUID-FLOW MODEL

Our purpose is to ensure that any session is served above its floor rate, provided that adequate admission control is exercised, while simultaneously allowing all sessions to share available capacity with respect to their maximum rates. Once this is achieved, independent delay bounds can be obtained by requiring a specific amount of buffer. In this section we focus on the ideal model by making the fluid-flow assumption. We start by introducing some notations. For all sessions k , $k = 1, 2, \dots, N$, we denote by f_k the floor rates that are to be guaranteed and by m_k the maximum rates over which fairness is defined. An issue that arises is to choose whether fairness must be defined over m_k or over $m_k - f_k$, given that f_k is already guaranteed. From a practical standpoint, we consider defining fairness over m_k , and notice that the other option can be achieved merely by replacing m_k with $m_k - f_k$. $W_k(s, t)$

denotes the amount of traffic from session k served in the time interval $[s, t]$. The desired fairness is thus depicted by:

$$\frac{W_i(s, t) - (t - s)f_i}{W_j(s, t) - (t - s)f_j} \geq \frac{m_i}{m_j}, \quad (4)$$

holding for any session i that is backlogged throughout $[s, t]$. Summing (4) over all sessions j , we get

$$W_i(s, t) - (t - s)f_i \geq \frac{m_i}{\sum_{j=1}^N m_j} (t - s) [C - \sum_{j=1}^N f_j]. \quad (5)$$

Hence the rate allocated to session i at time t is:

$$x_i(t) = f_i + \frac{m_i}{\sum_{j \in \mathcal{B}(\tau)} m_j} \left[C - \sum_{j \in \mathcal{B}(\tau)} f_j \right] \geq x_i = f_i + \frac{m_i}{\sum_{j=1}^N m_j} \left[C - \sum_{j=1}^N f_j \right]. \quad (6)$$

The right hand part of (6), namely x_i , depicts the minimum service rate, which is guaranteed regardless of the behavior of other sessions. That is, the isolation property remains valid for this fluid-flow model. More, it is easily shown that using

$$\sum_{j=1}^N f_j \leq C \quad (7)$$

for admission control leads to an absolute guarantee of the floor rate. Such a guarantee allows for a guaranteed delay bound, provided that an adequate amount of buffer is dedicated to the session. Because of the isolation property, it turns out that this scheme can serve a variety of applications having different needs. For instance, a non-adaptive application requiring real-time service needs only set its floor rate to the bandwidth it needs, and request a null maximum rate, thereby reserving bandwidth for its whole stream. In contrast, an application being able to adapt to extreme congestion with no time constraint may indicate a null floor rate, hence only use whatever bandwidth is available. Between these two extremes, many combinations of guaranteed and shared bandwidth can be requested by applications requiring both. The isolation property ensures that any session gets its fair share, be it adaptive or not. As for the delay guarantee, it may be remarked that, similarly to GPS, sessions are guaranteed a worst-case delay provided their traffics conform to the allocated rates. This implicitly assumes that losses are not tolerated. But since adaptive applications, by nature, tend to shape their traffics to the available rate, so as to avoid losses, they actually tend to conform to their dynamically allocated rates, thereby getting delay guarantees. Moreover, by using segregated buffers, adaptive as well as non-adaptive applications incur losses depending only on their own behavior (such losses can be further corrected using *Forward Error Correction* techniques as applied to continuous data streams [1, 4]). Thus, following [15], we consider that apart from the lost packets these applications actually have guaranteed

delay bounds. We next focus on the actual fairness embedded in the model. Define $\rho(\tau)$ to be the *congestion ratio* (or theoretical loss rate) for the dynamically shared part of the bandwidth at time τ :

$$\rho(\tau) = 1 - (C - \sum_{j \in \mathcal{B}(\tau)} f_j) (\sum_{j \in \mathcal{B}(\tau)} m_j)^{-1}. \quad (8)$$

Let the incoming traffic at time τ be:

$$\sum_{j \in \mathcal{B}(\tau)} m_j = K(C - \sum_{j \in \mathcal{B}(\tau)} f_j) \quad K \geq 0. \quad (9)$$

A K greater than one depicts a congestion, and a K less than one indicates underutilization. From (6), we have, for any session i

$$x_i(\tau) = f_i + \frac{m_i}{K}, \quad (10)$$

hence

$$x_i(\tau) = f_i + m_i(1 - \rho(\tau)). \quad (11)$$

Equation (11) shows that any session is required to adapt its incoming traffic in the same proportion $\rho(\tau)$. This conforms to the envisioned fairness. Hence it turns out that such a scheduling model simultaneously answers the three issues that are raised by adaptive applications.

4 PACKET-BY-PACKET SCHEDULER

In this section, we consider actual packet networks, the nodes of which serve one packet at a time in a non-preemptable way. The beginning of our approach is almost identical to the packetized GPS scheme, except for the definition of the normalized service $w_k(s, t)$ received by a session k constantly backlogged during $[s, t]$, which comes from equation (6) above:

$$w_k(s, t) = \frac{W_k(s, t)}{x_k}, \quad (12)$$

where

$$x_k = f_k + \frac{m_k}{\sum_{j \in \mathcal{B}(s, t)} m_j} \left[C - \sum_{j \in \mathcal{B}(s, t)} f_j \right]. \quad (13)$$

We may define a virtual time function $v(t)$ representing the progress of work in the server, by:

$$v(t) - v(s) = w_k(s, t) \quad \forall k \in \mathcal{B}(s, t). \quad (14)$$

Multiplying each side of (14) by x_k and summing up over backlogged sessions k :

$$v(t)-v(s)\sum_{k \in B(s,t)}x_k = \sum_{k \in B(s,t)}x_k w_k(s,t), \quad (15)$$

hence

$$v(t)-v(s) = \frac{C(t-s)}{\sum_{k \in B(s,t)}x_k}. \quad (16)$$

By computing that

$$\sum_{k \in B(s,t)}x_k = C, \quad (17)$$

we conclude that our virtual time function is indeed the real time.

Now, focusing on the i 'th packet from session k , i.e. p_k^i , we notice that the beginning of its service is

$$b_k^i = \max(a_k^i, d_k^{i-1}). \quad (18)$$

The finishing time of p_k^i is thus

$$d_k^i = \frac{L_k^i}{x_k} + \max(a_k^i, d_k^{i-1}). \quad (19)$$

Equation (19), which depicts the iterative algorithm for computing finishing times (i.e. tags) of successive packets from session k , is indeed very close to Zhang's *Virtual Clock* scheme [32]. However in our case, the x_k values are not constant over time, but depend on the set $\mathcal{B}(t)$ of backlogged sessions at time t . Hence, the issue that remains is that of computing the set $\mathcal{B}(t)$, so as to compute the values for $\sum_{k \in \mathcal{B}(t)}f_k$ and $\sum_{k \in \mathcal{B}(t)}m_k$ at the time of tagging a new packet. We could just argue that the virtual time function of the *Packetized Generalized Processor Sharing* (PGPS, [26]) scheme also involves such a computation, hence our proposal requires very few additional work, and has the same complexity. However, some of the most important contributions to the fair queuing area were aimed at removing this computation, because it is considered too complex to be actually implemented (see for instance [20]). For this reason, we will discuss in the next section, which deals with implementation, one possible algorithm that reduces this complexity.

5 IMPLEMENTATION

In the following, we focus on the multi-queue implementation of GPS-like schedulers, i.e. each session traffic is stored inside a FIFO queue, and tags are computed only for the first packet of each backlogged session. As a side note, consider that the multi-queue version is advocated in several papers dealing with hardware implementation of the GPS scheme (e.g. [18, 27]). As exemplified in [19], determining the set of backlogged sessions at any time may be computationally intensive,

because, by the time a single packet is served, all sessions may potentially become backlogged or cease to be backlogged. For this reason, we restrict to approximate knowledge which allows for a greedy approach.

Let F and M be approximated values for $\sum_{k \in \mathcal{B}(t)} f_k$ and $\sum_{k \in \mathcal{B}(t)} m_k$, respectively. Each time a packet from session k is picked up for service, let

$$\begin{aligned} F &\leftarrow F - f_k \\ M &\leftarrow M - m_k \end{aligned}.$$

Similarly, each time a packet becomes first inside session k 's FIFO queue (either because the previous packet has been picked up for service, or because the session is newly backlogged), let

$$\begin{aligned} F &\leftarrow F + f_k \\ M &\leftarrow M + m_k \end{aligned}.$$

before computing its tag. It immediately follows that each time a session ceases to be backlogged its floor rate and maximum rate values are removed from F and M , respectively. Conversely, these values remain taken into account for a backlogged session. Nevertheless, a trouble may arise as a consequence of such a greedy approach. Consider the packet p_k^i from session k , being tagged at time s and picked up for service at time t . During $[s, t]$, several sessions may become backlogged or cease to be backlogged. Such changes will not be reflected by the tag of p_k^i . Hence the order in which packets depart the server may be different to the order in which they would have been served by the fluid-flow server (in the following, we call this phenomenon an *inversion problem*). Consequently we may wonder if the packet receives proper service. We answer this question below by assessing the worst-case delay bound of our scheme, with respect to the theoretical (i.e. fluid-flow) model. We first focus on a similar, albeit simpler, tag computation algorithm, in which tags are iteratively given by:

$$d_k^i = \frac{L_k^i}{\phi_k (\sum_{k \in \mathcal{B}(t)} \phi_k)^{-1} C} + \max(a_k^i, d_k^{i-1}), \quad (20)$$

where ϕ_k is the weight of session k in the GPS definition, and where the summation over $\mathcal{B}(t)$ is approximated by the greedy method above. This is actually an approximation of GPS. We now conduct a worst-case study for the delay bound of this scheme. Consider $N + 1$ sessions, numbered $1, 2, \dots, N+1$, so that

$$\sum_{k=1}^{N+1} \phi_k = C. \quad (21)$$

holds. Since we are looking for a worst-case delay bound, we must and will hereaf-

ter consider that all sessions are backlogged. Initially, only session 1 is backlogged and has a packet in service. During the service of this single packet, we make the worst-case assumption that all other sessions become backlogged, and we focus on the delay experienced by session $N+1$, when its packet is picked up last, contrary to the fluid-flow model in which it would be served first (i.e. an inversion problem happens). Indeed, we actually look for the maximum difference between the actual delay and the theoretical delay, as given by the fluid-flow model. Without loss of generality, we make the following simplifications: All sessions i belonging to $1, \dots, N$ have the same weight ϕ_i . All packet sizes are equal to the maximum packet size L_{max} . We also take $C = L_{max}$ to clarify the computations. We start at time $t = 0$, and all sessions in $2, \dots, N+1$ become backlogged one after the other, in order. The service of the packet from session 1 has a finishing time equal to 1. Hence the tag for session 1 (i.e. for the packet in its FIFO queue) is $T_1 = 1$ (for its service time) + $\max(1, 0) = 2$. When session 2 becomes backlogged, its tag is also 2, but now because the weight summation (as approximated by the greedy approach) equals 2. The same applies to any session i up to N , receiving a tag $T_i = i$. Now, considering session $N+1$, we look for its weight such that an inversion problem arises. For this to happen, its tag must be greater or equal to the highest tag already computed in the system, that is:

$$T_{N+1} \geq N, \quad (22)$$

hence (from 20)

$$\frac{L_{max}}{\phi_{N+1}(N\phi_i + \phi_{N+1})^{-1}C} + \max(0, 0) \geq N. \quad (23)$$

From (21), since all sessions are backlogged

$$\phi_{N+1} \leq \frac{C}{N}. \quad (24)$$

We select the highest possible value for ϕ_{N+1} so as to minimize the fluid-flow delay (thereby maximizing its difference with the actual delay). Then, the fluid-flow delay it experiences is:

$$D_{GPS} = \frac{L_{max}}{\phi_{N+1}} = N \frac{L_{max}}{C}. \quad (25)$$

The actual delay it experiences is the time it takes to serve one packet from all sessions, since session $N+1$ is served last because of the inversion problem. Hence

$$D_{actual} = (N+1) \frac{L_{max}}{C}. \quad (26)$$

It turns out that the difference between fluid-flow delay and actual delay is the time it takes to serve one packet of maximum size at the server rate. Since this is a worst-case value, in the general case our implementation ensures the following delay bound, for the i 'th packet of any session k :

$$d_k^i - a_k^i \leq \frac{L_k^i}{\phi_k} + \frac{L_{max}}{C}. \quad (27)$$

Equation (27) is well known to depict the optimal delay bound for packet-by-packet GPS-like schedulers [26]. For instance it is achieved by the PGPS scheme introduced in [26], which has, however, an overall complexity of $O(N)$. Conversely, the *Self-Clocked Fair Queuing* proposal ([19, 20]) has a low complexity of $O(\log N)$, which makes it appealing for broadband implementations, but suffers from a sub-optimal delay bound, where the discrepancy between theoretical and actual delay can be as large as N times the optimal value [21]. We actually proposed in (20) above a new implementation of GPS, which combines reduced complexity of $O(\log N)$ with the optimal delay bound. A third criterion relates to the maximum discrepancy between the service received by any two sessions, as studied in [2, 19]. We state without proof that the scheme depicted by (20) is within a factor two of the minimal discrepancy. The proof will be released in a forthcoming paper. Turning back to our previous scheduler integrating guaranteed floor rates and maximum rate fairness, we immediately get its delay bound by merely reminding that, for the worst-case to happen, all sessions must be backlogged and the guaranteed rate summation must equal C . While the guaranteed rates were depicted by ϕ_k values in (20), they are now given by the different floor rates f_k . In such a case, it follows from (6) that $x_i = f_i$. As a consequence, the worst-case study for this scheme can be conducted merely by replacing ϕ_k with f_k in the above demonstration. Since two summations over $\mathcal{B}(t)$ are to be approximated instead of one, it turns out that our proposal also has both a low complexity and the optimal delay bound.

6 CONCLUSION

We have proposed in this paper a new scheduling policy of the Generalized Processor Sharing family, which provides an integrated solution to the enforcement of quality of service guarantees for adaptive multimedia applications. These QoS guarantees relate to three major issues: floor rate guarantee, bounded delay for all interactive applications, and need to enforce adequate fairness. We have shown that our proposal, by providing isolation, makes it possible for different adaptive (and also non-adaptive) applications to harmoniously share network resources, based on a fairness definition which is both a common-sense one, and a well accepted one (i.e. proportional fairness). We also proposed a low complexity implementation (i.e. amenable to broadband speeds) that provides the optimal delay bound for a packet GPS-like scheduler. The simplified example we studied is actually an implementation of GPS, with the same complexity as the *Self-Clocked Fair Queuing* scheme

proposed by Golestani in [19], and a better delay bound. In addition, we point out that QoS routing (which is still largely an open issue) could indeed use the F and M values as indicators of the actual node occupancy (although no effective congestion takes place, due to applications adapting to available bandwidth), so as to select the least loaded paths. These indicators could also be of great value for answering network dimensioning issues. But the remaining issue we believe is the most important relates to the maximum rate values, that must be provided to the network for proper fairness enforcement. It is user cooperation, as it is required that users tell the truth in stating their maximum rate for fairness to be consistent. Walking into the steps of Bolot in [3], Kelly in [23], and Shenker in [30], we believe that a workable way to enforce this is to make use of pricing as an incentive not to overestimate one's needs. Indeed, guaranteed floor rates could be charged a high price (depending on bandwidth reservation), in order to discourage overprovisioning, and, conversely, shared bandwidth could be charged a lower price related to maximum rates, that is, independent of the actual rates achieved by applications. The latter price would reflect relative bandwidth purchases rather than absolute ones. We believe that such an idea is close to the one advocated by MacKie-Mason and Varian in [25], where an auction process for bandwidth allocation is presented. In addition, consider that such a pricing scheme ultimately gives an economic meaning to the fairness that the network enforces.

The author is fully indebted to Maher Hamdi, Laurent Toutain, Pierre Rolin and Alain Léger for their valuable support. Also thanks to the anonymous reviewers for their insightful comments. This work was done within the CNET-DEST collaboration number 93 PE 7301.

7 REFERENCES

- [1] A. Albanese, J. Blömer, J. Edmonds, M. Luby. *Priority Encoding Transmission*. ICSI Technical Report TR-94-039, Berkeley, 1994.
- [2] J.C.R. Benett, H. Zhang. *WF²Q: Worst-Case Fair Weighted Fair Queueing*. In Proceedings of IEEE Infocom, pp 120-128, San Francisco, CA, march 1996.
- [3] J-C. Bolot. *Cost-Quality Tradeoffs in the Internet*. To appear in Computer Networks and ISDN Systems.
- [4] J-C. Bolot, A. Vega-García. *Control Mechanisms for Packet Audio in the Internet*. Procs. IEEE INFOCOM'96, pp 232-239. San Francisco, march 1996.
- [5] J.C. Bolot, T. Turetti. *A Rate Control Mechanism for Packet Video in the Internet*. Procs. IEEE Infocom'94, pp. 1216-1223, Toronto, Canada, June 1994.
- [6] R. Braden, D. Clark, S. Shenker, *Integrated Services in the Internet Architec-*

ture: an Overview. Request for Comments 1633, june 1994.

- [7] A. Campbell, G. Coulson. *Experiences with an adaptive multimedia transport system in a QoS Architecture*. Technical Report, Columbia University, available at URL <http://www.ctr.columbia.edu/~campbell/papers/adapt.ps.gz>
- [8] D.D. Clark, S. Shenker, L. Zhang. *Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism*. Proc. ACM SIGCOMM'92, august 1992.
- [9] C. Compton, D. Tennenhouse. *Collaborative Load Shedding for Multimedia Based Applications*. Int. Conference on Multimedia Computing Systems, Boston. may 1994.
- [10] L. Delgrossi *et al.* *Media Scaling for Audiovisual Communications with the Heidelberg Transport System*. Procs. ACM Multimedia'93, Anaheim, California, august 1993.
- [11] A. Demers, S. Keshav, S. Shenker. *Analysis and simulation of a fair queueing algorithm*. In Journal of Internetworking Research and Experiment, pp 3-26, october 1990. Also in Procs. ACM SIGCOMM'89, pp 3-12.
- [12] C. Diot, A. Seneviratne. *Quality of Service in Heterogeneous Distributed Systems*. Submitted to HICSS-30. Hawai. January 1997.
- [13] C. Diot, C. Huitema, T. Turletti. *Multimedia Applications should be Adaptive*. HPCS Workshop. Mystic (CN), August 23-25, 1995.
- [14] C. Diot. *Adaptive Applications and QoS Guaranties*. IEEE Multimedia Networking Conference. Aizu (Japan). September 27-29, 1995.
- [15] D. Ferrari. *Client Requirements for Real-Time Communication Services*. RFC 1193, also in IEEE Communications Magazine vol 28(11), november 1990.
- [16] A. Fladenmuller, A. Seneviratne, E. Horlait. *A Hybrid QoS Management Scheme for distributed multimedia applications*. Procs. PROMS'95 workshop, Slazburg, Austria, october 1995.
- [17] S. Floyd, V. Jacobson. *Random Early Detection Gateways for Congestion Avoidance*. IEEE/ACM Transactions on Networking, vol 1, no 4, pp 397-413, august 1993.
- [18] M.W. Garrett. *A Service Architecture for ATM: From Applications to Scheduling*. IEEE Network, May /June 1996, pp 6-14.
- [19] S. J. Golestani. *A Self-Clocked Fair Queueing Scheme for Broadband Applications*. In Procs IEEE INFOCOM'94, pp 636-646, Toronto, CA, june 1994.

- [20] S. J. Golestani. *Fair Queueing Algorithms for Packet Scheduling in BISO.* Procs. IZS'96, Zürich, 1996.
- [21] P. Goyal, S. Lam, H.M. Vin. *Determining End-to-end Delay Bounds in Heterogeneous Networks.* in Procs. NOSSDAV'95, Durham, april 1995.
- [22] P. Goyal, H.M. Vin, H. Cheng. *Start-time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks.* in Procs. ACM SIGCOMM'96, Stanford, CA, august 1996.
- [23] F. Kelly. *Charging and rate control for elastic traffic.* to appear in European Transactions on Telecommunications, January 1997.
- [24] C. Lefelhocz, B. Lyles, S. Shenker, L. Zhang. *Congestion Control for Best-Effort Service: Why We Need a New Paradigm.* IEEE Network vol 10(1), january / february 1996.
- [25] J. MacKie-Mason, H. Varian. *Some Economics of the INTERNET.* 10th Michigan Public Utility Conference, march 1993.
- [26] A.K. Parekh and R.G. Gallager. *A Generalized Processor Sharing Approach to Flow Control In Integrated Services Networks-The Single Node Case.* In Proceedings of Infocom, pages 914-924. IEEE, 1992.
- [27] J.L. Rexford, A.G. Greenberg, F.G. Bonomi. *Hardware-Efficient Fair Queueing Architectures for High-Speed Networks.* Procs. IEEE INFOCOM'96, San Francisco, 1996, pp 638-646.
- [28] S. Shenker. *Fundamental Design Issues for the Future Internet.* IEEE Journal on Selected Areas in Communications, vol. 13 (no. 7): 1176-1188, Sep. 1995.
- [29] S. Shenker. *Making Greed Work in Networks: A Game-theoretic Analysis of Switch Service Disciplines.* Procs. SIGCOMM'94 Conference, Oct. 1994, ACM, New York, pp. 47-57.
- [30] S. Shenker. *Service Models and Pricing Policies for an Integrated Services Internet.* in Public Access to the Internet, B. Kahin and J. Keller eds., MIT Press.
- [31] J. Wroclawski. *Specification of the Controlled-Load Network Element Service.* Internet Engineering Task Force, INTERNET-DRAFT draft-ietf-int-serv-ctrl-load-svc-02.txt, june 1996, work in progress.
- [32] L. Zhang. *Virtual Clock: A New Traffic Control Algorithm for Packet Switching Networks.* In Procs. ACM SIGCOMM'90, pp 19-29, Philadelphia, PA, september 1990.

Adaptive Variation of Reliability

Robin Kravets, Ken Calvert, P. Krishnan and Karsten Schwan*

*Georgia Institute of Technology, Bell Labs**

College of Computing

Georgia Institute of Technology

Atlanta, Georgia, USA

email: {robink,calvert,schwan}@cc.gatech.edu

Bell Labs

Holmdel, New Jersey, USA

email: pk@research.bell-labs.com

Abstract

This paper presents the specification and implementation of a variable reliability framework. By using this framework, applications can explore interesting performance enhancing techniques. Some of these techniques include the use of quality-based partitioning of data and the use of adaptation of reliability based on the current state of the application and the network. We present results from an initial implementation and demonstrate the usefulness of adaptive reliability variation for sample distributed applications.

Keywords

adaptive communication, variable reliability, sliding window protocol, data partitioning

1 INTRODUCTION

Historically, applications had to choose between completely reliable message transfer (as with TCP) and “best effort”, non-guaranteed service (as in UDP). This presents little choice to applications able to handle relaxed levels of reliability. Instead, applications may have to pay the price of using a reliable protocol, which can be measured in the unnecessary retransmission of messages by the sender and in the extended buffering of messages by the sender and the receiver. Alternatively, applications may use unreliable protocols, which may result in unacceptable losses.

The research contribution described in this paper is a framework for implementing variable reliability communication. The use of this framework provides applications with opportunities to realize performance gains by exploiting tradeoffs with respect to the reliability of message communication. Specifically, with the framework, an application can express its data’s reli-

ability requirements more precisely. This information can then be combined with feedback from the network resources being used. The application can use such feedback to adjust its requirements during execution, thereby improving its use of available network resources. Preliminary, experimental results demonstrate that this approach leads to improved application performance compared to using reliable communications.

This research is motivated by multimedia applications such as medical imaging, distributed virtual environments, and distributed simulations. The service requirements of such applications range from offering transmission times matching human perceptual needs (as in distributed virtual environments) to providing suitable reliability at acceptable speeds (as in medical imaging). In addition, any single application is likely to communicate via multiple data types, resulting in the simultaneous specification of multiple service requirements.

The remainder of this paper is organized as follows. Section 2 describes the variable reliability framework, and it presents a simple yet powerful way in which applications can specify their diverse reliability requirements. Section 3 demonstrates the use of intelligent partitioning of application data in conjunction with the use of variable reliability. We next present a generalized cost framework in which the application can specify service tradeoffs as payoff functions. This generalization is studied in Section 4, in which we show how these payoff functions may be used to adapt communication strategies to changes in communication patterns. Section 4.3 presents simple adaptation algorithms and demonstrates experimentally that the use of a variable reliability protocol allows applications to effectively adapt to changing communication needs. Concluding remarks appear in Section 5.

2 A VARIABLE RELIABILITY FRAMEWORK

It is well-known that for multimedia applications the two types of reliability provided by TCP and UDP are insufficient. This is due to the fact that most such applications are able to tolerate some application-specific losses. Given a lack of choice, such applications must use reliability even when they do not need it or implement their own versions of reliability on top of an existing unreliable protocol. Marasli, Amer and Conrad (Marasli *et al.* 1996) quantify the cost of “too much” or “too little” reliability within the specification of their protocol. Similarly, Delgrossi, et al. (Delgrossi *et al.* 1993), evaluate the cost of using standard reliable sliding window protocols like go-back-*n* and selective retransmission in situations where reliability could be relaxed. In comparison, our approach is to provide a framework in which each application can define the notion of “reliability” it needs.

Implicit in our design is the concept of *application layer framing* (ALF) (Clark *et al.* 1984). Namely, we assume that an application can deal independently with the data units that it sends. In other words, each unit of data sent

by the application contains sufficient identifying information to allow the application to process it. Among other things, this means that data need not be held up for ordering constraints (beyond those defined by application framing boundaries – as explained below).

Reliable data transfer can be defined as a service that guarantees to deliver data sent from a sender to a receiver without duplication, loss, or messages delivered out of order. Unreliable data transfer makes no guarantees as to duplication, loss, or order. However, it is not obvious how to define a service “between” these extremes. The following sections discuss the issues involved in defining such a service and describe a specific implementation of a variable reliability protocol.

2.1 Specification of Variable Reliability

Variable reliability mechanisms can be applied to both realtime and non-realtime data. This paper focuses on the issues for non-realtime data (i.e., data that is time-sensitive, but not time-critical). Work by Delgrossi, et al. (Delgrossi *et al.* 1993) and Papadopoulos and Parulkar (Papadopoulos *et al.* 1996) addresses reliability for realtime continuous traffic streams. By concentrating on non-realtime data, we can ignore lifetime issues for individual messages, and concentrate on issues about the amount of data received and the spacing of message losses. We target applications where the structure of the data is important and where timing issues are based around frame transmission time, not individual message transmission time. This type of specification can be particularly useful for applications running over low-bandwidth and/or high error rate networks (e.g., wireless networks).

The scheme we specify below defines a service based on simple loss measurement and retransmission policies. The protocol specified by Marasli, Amer and Conrad (Marasli *et al.* 1996) provides probabilistic reliability guarantees based on the number of retransmissions of a message. In contrast, our protocol provides applications with hard guarantees about reliability based on specified loss allowances. The following sections describe some parameters of our variable reliability protocol.

(a) Loss Tolerance

Two mechanisms govern loss tolerance for our variable reliability protocol. The first is the definition of *application-specified data boundaries* or *data frames*. A data frame consists of data messages, each of which is an *application data unit* (ADU). The messages from different frames are handled separately, thereby limiting the effects of problems in the transmission of one frame on the transmission of others. The second mechanism is a simple *sliding window* within the messages of the data frame. The use of these mechanisms is controlled by the specification of suitable loss tolerance parameters.

Two parameters may be specified by the application at the data frame level. The first parameter is the *maximum number of consecutive losses*, which essentially defines the tolerable ADU loss burst size. The second parameter is the *high level loss percentage*, which indicates how much loss in the total data frame the application can tolerate. At the sliding window level, the application can define the *maximum number of losses allowed in a window*. These parameters are similar to those specified by Gong and Parulkar (Gong *et al.* 1992).

In Figure 1, the maximum number of consecutive losses is 2, which means that there will never be more than two consecutive losses; the high level loss percentage is 50%, which means that the receiver is guaranteed to receive at least 50% of the ADUs in each frame; and the maximum number of allowable losses in a window of 6 is 3. The combination of the maximum number of allowable losses and the maximum burst size ensures that the losses are spread out through the messages of the data frame and not grouped together in one portion of the frame.

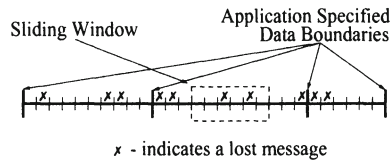


Figure 1 Example of Allowable Loss in a Sliding Window and in a Data Frame

(b) Application-Specified Data Boundaries

The application can specify how many data frames it can handle simultaneously. The protocol will then guarantee that the data being passed to the application will always belong to appropriate frames. Given that some number n , of frames may be transmitted simultaneously, we define a sliding window on data frames. If the sliding window size is n , at most n frames will ever have ADUs in transmission at any given point in time. The sender will not start to send the $n + 1$ st frame until the first frame has been “successfully” transmitted, where “successfully” is defined in accordance with the specified reliability. Messages from old frames will be discarded on the receiving side.

2.2 Variable Reliability Protocol

The use of a sliding window mechanism drives the protocol’s implementation. In a sliding window protocol, decisions on when to ask for specific retransmissions are made by the receiver. These decisions are made based on the policy

defined for the specific protocol. For example, the policy for a reliable protocol would be to ask for a retransmission of all messages determined as lost by the receiver. In such a protocol, an ACK indicates that the message has been received successfully by the receiver. But this definition is a policy decision. For our purposes, we would like to say that an ACK indicates that the sender no longer needs to buffer or retransmit the ACKed message; thus, depending on the policy used in a given protocol, the sender may receive ACKs for messages that were never received. The key is that the receiver controls the ACK policy.

In our variable reliability protocol, the receiver tracks messages using two information sources: the receive window and the receive history. The receive window indicates the range of *active* messages, where an active message is a message that may still be accepted and buffered by the receiver. Information about the reception of messages is maintained in this window in order to prevent the passing of duplicate messages to the application. The receive history is used to determine if a loss in the receive window can be allowed; messages in the receive history are no longer active, and they will not be accepted by the receiver. This is because message losses in the receive history have already been “allowed”. Additionally, the receiver assumes that messages received out of order are lost. This assumption permits simplification of the protocol at the cost of some unnecessary losses and retransmissions.

The protocol uses cumulative ACKs and NAKs. When the receiver detects a lost message, it checks to see if that loss violates any loss tolerance parameters. If so, the receiver sends a NAK for that message. If not, the message will be acknowledged in the next timer-generated ACK. Due to the fact that the loss of the last message in a window or frame may go unnoticed by the receiver, the sender maintains a timer for each message, which, on expiration, causes the message to be retransmitted. A number of techniques may improve protocol performance for this kind of occurrence, including indicating the number of messages in a data frame or sending markers after the last message in a data frame.

Figure 2 shows an example of the receive window and the receive history during execution. In this example, the receive window size is 10 and at most 3 messages may be lost out of any 10 consecutive messages. The maximum number of consecutive losses is 1. The variable *nextExp* is the sequence number following the highest-numbered received message. The variables *waitPoint* and *tooFar* delimit the receive window. The value of *waitPoint* indicates the message that the receiver is currently waiting for. If the receiver is not waiting for any outstanding messages, then *waitPoint* is set to *nextExp*. *tooFar* is always *waitPoint* plus the receive window size. In the case of an unacceptable loss, *waitPoint* will be set to the sequence number of that message. The variable *holdPoint* indicates the oldest lost message in the receive history. This and any other later allowed losses in the receive history will determine if a

new loss can be allowed. *holdPoint* is at least *waitPoint* minus the receive window size.

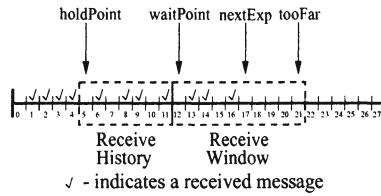


Figure 2 Window Parameters for a Variable Reliability Protocol

This snapshot shows the receiving protocol waiting for message number 12. In this example, the receive history indicates that messages 5, 7 and 10 were allowed to be lost. On the receipt of message 13, the receiver determines that it cannot allow message 12 to be lost, because 3 messages from the “window” (i.e., 3-12) have already been lost. The receiver sends a NAK for message 12. While it is waiting for the retransmission, it continues to process the new incoming messages, and so advancing *nextExp* to 14, 15 and then 17. Once the retransmission is received, *waitPoint* is advanced to message 15, and *holdPoint* is advanced to message 7.

This section has described the definition for a specific reliability stream. In order to enable applications to switch between different reliability levels, the variable reliability framework can maintain multiple reliability streams in parallel. The sender can specify any one of these streams for each data message it sends. In this way, the sender determines what reliability is required for specific messages, and the receiver implements the policies for each reliability stream.

3 QUALITY-BASED PARTITIONING OF DATA

Applications may handle data that can be grouped into different levels of quality. Quality may refer to running time, reliability or any such service parameters. For this discussion, we equate quality with reliability. In section 4, we will investigate the effects of multiple quality parameters. Essentially, quality is defined specifically for each application. In some situations, quality may even differ among the multiple messages of an application’s single data frame. In order to ensure that the requirements of the data with the strictest requirements are met, the application must pay the price for processing all of the data at that quality level. By allowing the application to more precisely define the quality requirements of its data, the communication can be adapted accordingly.

We illustrate the concept of quality-based partitioning of data and its in-

teraction with our variable reliability mechanism using two distinct examples. Our first example explores static quality-based partitioning with an image transfer application that requires different reliability levels for fixed regions of the image. Our second example explores dynamic quality-based partitioning with a distributed simulation that dynamically changes the reliability requirements of regions of its world.

3.1 Image Transfer Application

Image transfer provides an excellent avenue for data partitioning. In image transfer, each image is a data frame. If we consider the transfer of an image of a person, the most important part of that image might be the face (see figure 3(a)). This section of the image is defined with a strict quality, while the quality of the rest of the image is allowed to decrease as we move further away from the face.

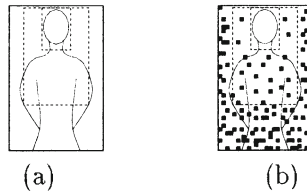


Figure 3 Image with Application Specified Quality Regions

We can now combine the concept of the variable reliability protocol with quality-based data partitioning. Through this combination, we can ensure that the face is received at one hundred percent reliability, while the reliability levels of the rest of the image decrease as we move away from the face. (see figure 3(b), black spaces indicate lost messages.)

For our experiments, we use an application that transfers a 6.5Mbyte image across a 10Mb Ethernet connection. The application runs under three different partitioning configurations. Configuration 1 imposes 100% reliability on the entire image. Configuration 2 partitions the image into two regions: 11% of the image (e.g., the face) requires 100% reliability; 89% requires 66% reliability. Configuration 3 partitions the image into three regions: 11% requires 100% reliability; 28% requires 66% reliability; and 61% requires 33% reliability. Each of these configurations is run at progressively higher message loss percentages. We simulate random loss of messages.

Figure 4 depicts the running times for the transfer of a partitioned image in a lossy network under each of these conditions. The results are an average of 10 runs at each loss percentage rate. The closer we come to specifying the reliability requirements of the application, the better we can judge the

necessity for message retransmission. As the loss increases, the running time for configuration 1 increases to 280% of the running time for a run with no losses. In comparison, configuration 2's running time increases to 180% and configuration 3's running time to only 136%. The improved transmission time in configuration 3 is gained while remaining within the reliability requirements of the application.

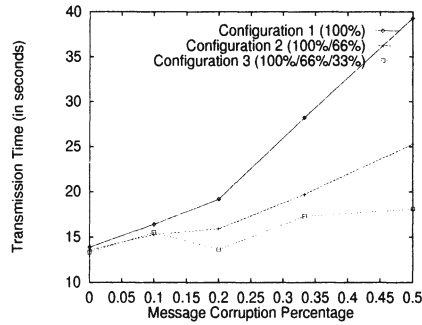


Figure 4 Transmission Time for Quality-Based Partitioned Image

3.2 Distributed Simulation Application

The second application used for protocol evaluation is a distributed game, simulating multiple robots acting on a joint playing field and sharing a world view. The world view in this application is processed as an image and passed back and forth between the robots as it is being updated. The world is decomposed into blocks, where each block has an owner that is responsible for coordinating reads and writes as well as sending updates to the other robots. As each robot moves through the world, it updates the data in the world and receives updates from other robots.

The quality-based partitioning of data and the variable reliability protocol lend themselves well to being used for this application, since each piece of data potentially has a different reliability requirement, and these requirements may change as the simulation evolves. Intuitively, each robot only cares about its immediate surroundings. In other words, the reliability of an update is determined by the update's proximity to the robot receiving the update. The sections that are out of the robot's view may only require periodic, unreliable updates, or no updates at all. The application can dynamically change the assignment of data to a specific reliability level. This allows the application to choose what data is most important to it, and pay the overhead of reliability for that data.

For this distributed simulation application, the world is divided into con-

centric “rings” around the location of the robot. Figure 5 shows an example from the viewpoint of robot 0. Each of these rings corresponds to a reliability level. The loss tolerance in the distant rings is higher than that of the closer rings. In other words, as the updates come from further away, the reliability of the update is relaxed.

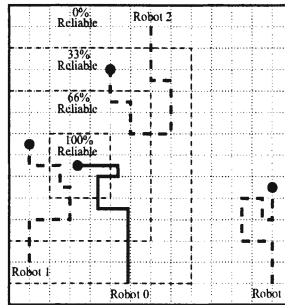


Figure 5 Snapshot of Distributed Robot Game with Variable Reliability Rings

The goal of this experiment is to determine the performance impact of allowing the robots to change the reliability of updates. Experiments monitor the number of retransmissions required and the number of allowed losses. Results are computed based on the number of updates received and the number of messages retransmitted. These numbers are averaged over four runs. Table 1 shows results for robot 0 from a configuration with 100% reliability and from a configuration with the type of partitioning described above. In runs using a totally reliable world, all losses triggered a retransmission request. In runs using quality-based partitioning, only 7% to 22% of the losses trigger retransmission requests. This percentage increases as the amount of reliability requested is increased. For a more detailed explanation of this experiment and the results, see (Kravets *et al.* 1996).

	Average Number of Updates Received	Average Number of Messages Retransmitted
100% Reliability	15377.00	5058.25
Quality-Based Reliability	11838.00	1728.00

Table 1 Average Updates Received and Messages Retransmitted (Robot 0)

4 ADAPTATION OF COMMUNICATION REQUIREMENTS

One of the most demanding requirements of distributed applications is that they continue working under unpredictable network conditions. Not only must they continue working, the applications must also maintain certain characteristics. Examples of these characteristics are response time, image quality, or speech comprehension. This may be difficult if the application considers the communication to be a static entity. The ability to adapt to current network conditions can make the difference between a usable and an unusable application (Diot *et al.* 1995). In this section, we consider the concepts involved in communication adaptation and study the effects of different adaptation methods, focusing on reliability. Our ongoing work deals with intelligent variation of other parameters like flow control, acceptable compression, etc.

Communication parameters are monitored and updated by a communication controller. The design of our experiments allows for differing controllers. One model may have the control centered in the application, thereby localizing the control to changes in that application. Another model includes an external controller, which may be able to control multiple applications. Our goal is to allow each application to determine what type of control model best suits it. In our applications, the controller is implemented as a library of functions used by the application.

4.1 Adaptation Measurements

As network service characteristics change, applications can exploit the variable reliability mechanism to adapt to these changes. In order to determine the benefit of using adaptation techniques, we need to quantify the benefits. By introducing the use of payoff functions, the application can assign priorities to different types and levels of quality. In general there is a payoff function, P_v , associated with each “variable”. These functions are specified by the application. The net payoff, P_{total} , is a function, Q , of all the defined P_v . For our current measurements, we have two variables: transfer time and reliability. Their payoff functions are $P_{\text{time}}(\text{TransferTime})$ and $P_{\text{reliability}}(\text{Reliability})$. Net payoff is defined as $P_{\text{total}} = Q(P_{\text{time}}(\text{TransferTime}), P_{\text{reliability}}(\text{Reliability}))$.

Williamson and Cheriton (Williamson *et al.* 1991) define the concept of loss-load curves to determine the action of the application in the face of congestion. Given an application’s transmission rate and the current state of the network, the loss-load curve provides the application with the percentage of packets that may be dropped. This technique allows the application to determine the cost of high transmission rates with higher loss rates, but places the responsibility of dealing with losses in the hands of the application. In contrast, we supply a mechanism for handling losses and allow the application to

specify its parameters. A controller can make use of information provided by loss-load curves to help it determine the best course of action.

Sample payoff functions appear in figure 6. As we would expect, the payoff for reliability is highest when one hundred percent of the data is received. This slowly drops off until it reaches the limit of acceptable reliability, and at this point drops to zero. Similarly, the payoff for time is highest for shorter transmission times, and drops to zero for some maximum acceptable transmission time.

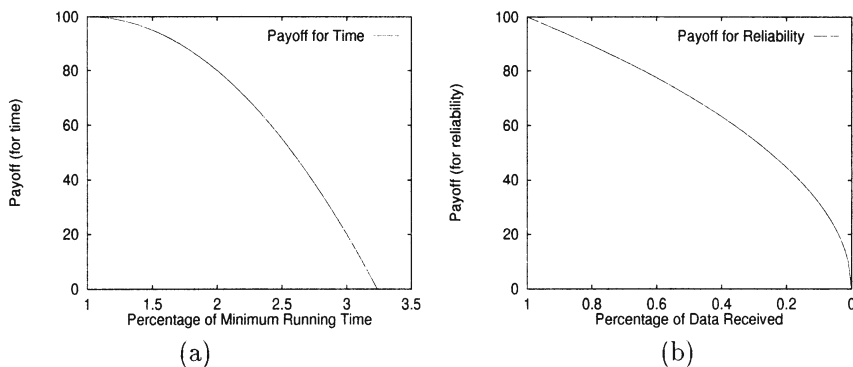


Figure 6 Actual Payoff Functions for Time and Reliability

4.2 Adaptation Locality

By assessment of available resources and of the quality requirements provided by the application, the communication can be adapted to better fit the application's needs and the available resources. Adaptation locality answers the question of where resource information comes from. A controller will have many different sources of information from which it can determine its current state. We define three levels of adaptation locality. A controller using *local adaptation* methods only considers information about the current running state of a piece of an application. For example, a controller may only know about the sending part of the application. The applications in section 3 show the use of local adaptation methods. A controller using *end-to-end adaptation* considers information from endpoints of an application. For end-to-end adaptation, we would have communicating controllers at the sending and receiving sides of an application. The experiments in section 4.3 consider end-to-end adaptation. A controller using *global adaptation* methods considers information from local and external resources. In the case of global adaptation, many different controllers may exchange information. Our ongoing work considers the impact of global adaptation.

4.3 Experiments

This section describes the results of two different experiments designed to explore the effects of end-to-end adaptation. Both experiments use a simplified version of the image transfer application described in section 3. In this version, the entire image is set to be at one reliability level. These experiments compare the effects of two different adaptation algorithms.

For both experiments, the controller is on the sending side and determines a course of action using local information and information collected by the receiver and returned to the controller. Specifically, in response to changes in the error rate, the controller can change the reliability level being used. The current implementation gives the controller the ability to dynamically switch between four distinct reliability streams: 100% reliability allows no losses; 66% reliability allows 1 consecutive loss and 34% in a window; 33% reliability allows 3 consecutive losses and 67% loss in a window; and 0% reliability allows any loss it notices.

(a) Adaptation Algorithms

Adaptation algorithms define how resource information is used in determining the service provided to the application. We define two adaptation algorithms: threshold adaptation and payoff adaptation. This research investigates reactive adaptation methods. Future work will consider the use of predictive (Bihari *et al.* 1991) and learning-theory-based adaptation methods (Krishnan *et al.* 1995, Keshav *et al.* 1984).

Threshold Adaptation: A controller using threshold adaptation has a fixed scheme that it uses to value resource information. Whenever a predefined threshold is crossed, the controller automatically changes the communication. In this example, the controller monitors the message loss percentage, which is quantized in four ranges. Each range is associated with a reliability level. If the observed message loss percentage crosses a threshold from one range to the next, then the controller adjusts to the reliability associated with the new range. In our application, if the controller has been seeing 1 percent message loss, which then jumps to 10 percent, the controller would reduce the reliability level from 100 percent to 66 percent. The placement of the thresholds within the range of message loss determines the performance of the application.

Payoff Adaptation: A controller using payoff adaptation considers the effect of changes on application performance. Before making any changes, the controller compares the cost of running with the current state with the payoff of running in the changed state. For this purpose, the controller needs information about the inter-relationship between the variables (via a function $f : \text{Reliability} \rightarrow \text{Time}$, for example). The payoff adaptation technique allows us to balance application requirements like quality level and running time with resource availability.

Reliability measurements are based on the total percentage of data received.

Total payoff is defined as $P_{total} = P_{time}(\text{TransferTime}) * P_{reliability}(\text{Reliability})$. The payoff functions that we used are shown in figure 6. These payoff functions represent some features matching the application’s behavior; in particular, the reliability drops off at a relatively constant rate up to 50% reliability and drops quickly after that. The payoff function for time allows for some minimal delay at a high payoff, but drops off sharply after two times the expected transfer time. We are currently investigating other payoff functions, e.g., fast dropping exponential functions. The operating points for the controller were gathered prior to the final experiment.

(b) Observations

Figure 7 shows the final payoff for each of the adaptation algorithms considered in our work. The goal of the payoff adaptation method is to adaptively choose the best running point for the application. We can see in figure 7 that the curve for the payoff adaptation method is essentially an upper envelope for the other curves and outperforms them at various points. This illustrates two points. First, a simple adaptive algorithm can exploit the benefits provided by the variable reliability protocol infrastructure, and it can choose the correct operating points efficiently. In other words, it can correctly choose the reliability level at which to run. Second, end-to-end adaptation has the potential to provide applications with sufficient feedback to adjust their reliability. In our current implementation, controllers exchange their statistics periodically. We are currently investigating the effect of changing the rate of transfer of information between the controllers.

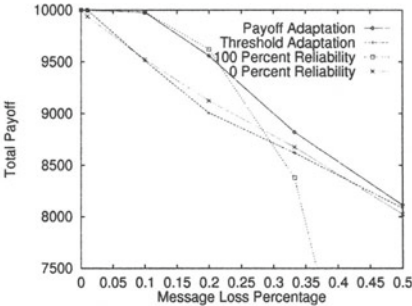


Figure 7 Final Payoff

It is illustrative to look at the raw running times of adaptation algorithms and to compare those times to their effectiveness in terms of the total number of bytes transferred. Figure 8(a) shows the running times for the two adaptation methods and two fixed reliability runs. The results show that for most of the experiment, the payoff adaptation method runs slower than the threshold adaptation method. If we consider the total amount of data that is received,

we can see in figure 8(b) that the payoff adaptation method also receives more of the transmitted data than the threshold adaptation method.

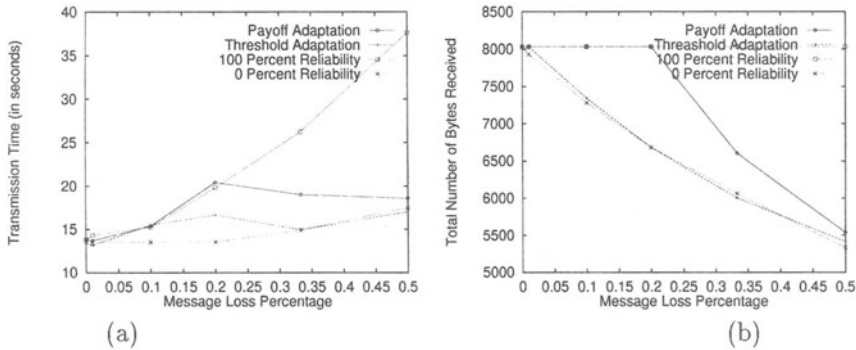


Figure 8 Time and Data Statistics

5 CONCLUSIONS AND FUTURE WORK

The premise of this work is that many multimedia applications will benefit from being able to choose between different reliability levels, rather than being forced to choose between full or no reliability. We have presented a mechanism by which applications can specify their varying reliability needs. We have implemented an infrastructure that provides these variable reliability options to applications. We have demonstrated that applications can exploit this framework to enhance performance in interesting ways; for example, by quality based partitioning of data and adaptively adjusting their reliability requirements to current network conditions. Our experiments show that significant improvements are possible by using the features of our variable reliability method.

Our ongoing work addresses other issues that fit into this framework. Specifically, we are looking at flow-control techniques, other adaptation methods via end-to-end communication of network information, allowing applications to choose other features (like compression level) and applicability on high-error/low-bandwidth networks, like wireless networks.

Acknowledgments

This work is sponsored in part by an AT&T PhD Fellowship and in part by NSF grants CDA-9501637, CDA-9422033, and ECS-9411846. Portions of this

work were completed at Bell Labs. We'd like to thank Binay Sugla for several helpful discussions and comments.

REFERENCES

- Bihari, T. and Schwan, K. (1991) Dynamic Adaptation of Real-Time Software. *ACM Transactions on Computer Systems*, **Vol. 9, No. 2**, 143-174, May 1992.
- Clark, D.D. and Tennenhouse, D.L. (1984) Architectural Considerations for a New Generation of Protocols. *Proceedings of the SIGCOMM '90 Symposium*, 200-208.
- Delgrossi, L., Halstrick, C., Herrtwich, R.G., Hoffmann, F., Sandvoss, J. and Twachtmann, B. (1993) The desirability of adjusting for residual effects in a crossover design. *First IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems (HPCS) '93*.
- Diot, C., Huitema, C. and Turetti, T. (1995) Multimedia Applications Should Be Adaptive. *Third IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems (HPCS) '95*.
- Gong, F. and Parulkar, G. (1992) An Application-Oriented Error Control Scheme for High Speed Networks. *Washington University, St. Louis, TR 92-37*.
- Keshav, S., Lund, C., Phillips, S.J., Reingold, N. and Saran, H. (1984) An Empirical Evaluation of Virtual Circuit Holding Time Policies in IP-over-ATM Networks. *Proceedings of IEEE INFOCOM 95*.
- Kravets, R., Calvert, K. and Schwan, K. (1996) Dynamically Configurable Communication Protocols and Distributed Applications: Motivation and Experience. *Georgia Institute of Technology, GIT-CC-96-16*.
- Krishnan, P., Long, P.M. and Vitter, J.S. (1995) Learning to Make Rent-to-Buy Decisions in Probabilistic Environments. *Machine Learning: Proceedings of the Twelfth International Conference*.
- Marasli, R., Amer, P. and Conrad, P. (1996) Retransmission-Based Partially Reliable Services: An Analytical Model. *Proceedings of IEEE INFOCOM 96*.
- Papadopoulos, C. and Parulkar, G. (1996) Retransmission-Based Error Control for Continuous Media Applications. *The 6th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV) '96*.
- Williamson, C.L. and Cheriton, D.R. (1991) Loss-Load Curves: Support for Rate-Based Congestion Control in High-Speed Datagram Networks. *Proceedings of the SIGCOMM '91 Symposium*.

On Compartmental Modelling of Multi-Service Communication Networks

M. Sivabalan, H.T. Mouftah and G. Takahara[†]

Department of Electrical and Computer Engineering

[†] Department of Mathematics and Statistics

Queen's University

Kingston, Ontario, Canada K7L 3N6

telephone : (613) 545-2934 fax : (613) 545-6615

e-mail : mouftah@eleceng.ee.queensu.ca

Abstract

Compartmental modelling, a technique that is widely used in areas such as bioscience, process control engineering, may serve as the basis for the performance evaluation of systems that are too complex to be handled by exact analytical/computational methods. With this technique, the dynamics of a system are described in terms of a set of differential or difference equations which permits the investigation of both the steady-state and transient behaviour of the system. In this paper, we describe how this technique can be used to model connection-oriented symmetric communication networks supporting multiple service classes, where each class is characterized by its bandwidth requirement, arrival and departure processes. We then demonstrate an application of this modelling technique to the steady-state analysis of a network and examine the accuracy of this modelling technique via simulations.

Keywords

Multi-service networks, network modelling.

1 INTRODUCTION

Compartmental modelling technique may serve as the basis for the performance evaluation of the systems that are too complex to be handled by exact analytical/computational methods. This technique is widely used in areas such as biology, medicine, and ecology. Recently, Garzia and Lockhart (1989) have studied its applicability in understanding the behaviour of connection-oriented communication networks supporting a single class of service. The results of this study indicate that compartmental models are computationally efficient and complex enough to capture the steady-state and transient behaviour of the networks. The goal of this paper is to show how this modelling technique can be applied to multi-service networks.

The rest of the paper is organized as follows: section 2 gives a brief review of compartmental modelling. Section 3 describes the multi-service network model that we consider in this paper. Section 4 explains the formulation of the compartmental model for the multi-service network. In section 5, we present and discuss the numerical results obtained using the compartmental model from the steady-state analysis of a network with two service classes. Section 6 assesses the accuracy of the compartmental model through comparison with simulations, and section 7 concludes this paper.

2 REVIEW OF COMPARTMENTAL MODELLING

Consider a system consisting of a finite number of distinct material types. In such a system, a *compartment* can be considered as a sub-system containing a single material type (Godfrey, 1983). In compartmental modelling, the system is visualized as an interconnected system of compartments. Compartments may exchange materials with each other and with the environment in accordance with the law of conservation. Transformation of a material type i into another material type j can be represented by a flow from compartment i to j . Also, the arrival of material type i into the system and the departure of material type i from the system can be represented by a flow from the environment to the compartment i and a flow from the compartment i to the environment respectively.

A system with n compartments can be described by the following differential equations:

$$\frac{dx_i}{dt} = f_{i0} + \sum_{\substack{j=1 \\ j \neq i}}^n (f_{ij} - f_{ji}) - f_{0i}, \quad (1)$$

where $i = 1, 2, \dots, n$, x_i is the amount of material in the compartment i and f_{ij} is the rate of flow from the *donor* compartment j to the *acceptor* compartment i , with 0 denoting the environment (see Figure 1).

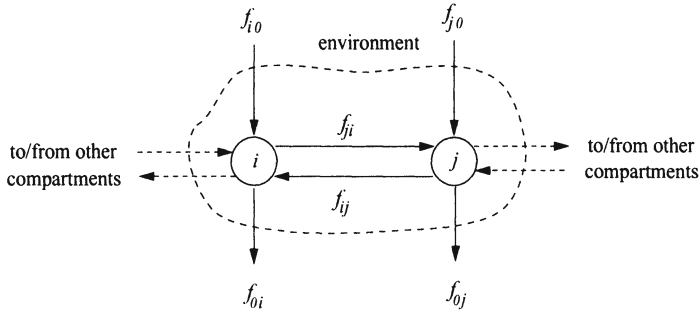


Figure 1 Two compartments of an n -compartment ($n > 2$) system.

In the simplest case, it is adequate to model the compartments as donor-controlled systems. In such systems, the rate of flow f_{ji} is related, linearly or nonlinearly, to the amount of material only in the donor compartment i . However, in a more complex model, the flow rate depends on the amount of material in not only the donor compartment but also in one or more other compartments. In general, the rate coefficients can be chosen to capture the non-linear, time-dependent, and state-dependent nature of a very complex system as follows:

$$\begin{aligned}
 f_{ij} &= a_{ij}(\mathbf{x})x_j(t) \\
 &\quad i = 0, 1, \dots, n \\
 &\quad j = 1, 2, \dots, n; j \neq i \\
 f_{i0} &= b_i(\mathbf{x}) \\
 &\quad i = 1, \dots, n,
 \end{aligned} \tag{2}$$

where $\mathbf{x} = (x_1(t), x_2(t), \dots, x_n(t))$ is the system state vector at time t , $a_{ij}(\mathbf{x})$ is the rate coefficient of the flow from the compartment j to i , and $b_i(\mathbf{x})$ is the flow rate from the environment to the compartment i (i.e., input). The equation 1 and 2 can be combined to form the following set of first-order differential equations:

$$\frac{dx_i(t)}{dt} = b_i(\mathbf{x}) + \sum_{\substack{j=1 \\ j \neq i}}^n (a_{ij}(\mathbf{x})x_j(t) - a_{ji}(\mathbf{x})x_i(t)) - a_{0i}(\mathbf{x})x_i(t), \tag{3}$$

where $i = 1, 2, \dots, n$. We refer to these equations as *state equations*. By solving these equations, we can obtain the steady-state as well as the transient behaviour of the system in terms of the quantity of interest \mathbf{x} .

Garzia *et al.* (1989) have applied the compartmental modelling technique to investigate the behaviour of a single service, circuit-switched networks. Because of the state-dependent nature of such networks, the resulting state equations are very complex and hard to solve. To overcome this difficulty, the authors have transformed the differential equation 3 into the difference equation 4 and analyzed the system in discrete-time domain.

$$\frac{x_i(t + \tau) - x_i(t)}{\tau} = b_i(\mathbf{x}) + \sum_{\substack{j=1 \\ j \neq i}}^n (a_{ij}(\mathbf{x})x_j(t) - a_{ji}(\mathbf{x})x_i(t)) - a_{0i}(\mathbf{x})x_i(t), \quad (4)$$

where $i = 1, 2, \dots, n$, and τ is the time-step. In section 4, we will show how the single service compartmental model developed by Garzia *et al.* (1989) can be extended to the symmetric multi-service network to be described in the following section.

3 NETWORK MODEL

Consider a fully connected symmetric network supporting multiple service classes. Each pair of network nodes is assumed to be connected by a pair of uni-directional links (one for each direction). Each service class is characterized by its bandwidth requirement, average arrival rate, and average holding time. We assume connection-oriented information transfer mode, e.g., ATM (de Prycker, 1991). In such a mode, upon arrival of a call, the network selects a path between the source and destination and reserves its resources along the chosen path. The information transfer for the call takes place on the chosen path. When the call is terminated, the resources reserved for that call is released. In this paper, we consider the link bandwidth as the only resource required for setting up calls, i.e., a call is blocked on a link if the link does not have enough bandwidth.

We assume that the link bandwidth is “quantized” into Basic Bandwidth Units (BBU's) and all links are of equal bandwidth. The bandwidth required to support a call from any class can be specified as an integral multiple of BBU. The calls are assumed to be bi-directional. We consider paths of at most two hops. The single and two hop paths are referred to as *direct* and *alternate* paths respectively. When a call set-up request arrives, the network first attempts to connect the call on the direct path. If this attempt fails, the network will try an alternate path to set up the call. When probing an alternate path, hops corresponding to that path are probed one after the other. We also assume that the time required to probe and establish a connection on a hop is constant, and both links corresponding to a hop are probed simultaneously.

The network is assumed to be spatially homogeneous, i.e., the network load is uniformly spread across the network. So, each link is offered an identical load. We further assume that the connection requests arrive at a link according to Poisson process. With these assumptions, the network blocking problem is reduced to the link blocking problem where the links block the calls independently (link independence assumption). Calls from a given service class are blocked with identical probabilities, and different classes could experience different blocking probabilities.

We will use the following notations for the network and call related parameters throughout the rest of this paper:

- N : number of nodes in the network.
- K : number of service classes supported by the network.
- C : link bandwidth in BBU's.
- τ : time to probe and establish connection on the pair of links between two nodes. This is what is going to be used as the time-step (see equation 4) in the state equations to be derived in section 4.
- b_i : bandwidth required (in BBU's) for a class i call.
- λ_i : external arrival rate (in calls/sec) of class i calls.
- $\frac{1}{\mu_i}$: average call holding time (in seconds) of a class i call.
- $\rho_i(t)$: link offered load at time t due to class i calls.
- $q_i(t)$: link blocking probability that class i calls experience at time t . Also, $p_i(t) = 1 - q_i(t)$.
- r_i : number of alternate paths between a source-destination pair used for class i calls.

4 FORMULATION OF THE COMPARTMENTAL MODEL

Between its arrival and departure, a call can be in a number of distinct phases (see Lee, Hluchyi and Humblet, 1995 for a detailed description of these phases). In this paper, we consider only two such phases. The first one is the probe phase in which the call is attempting to get a path, and the second one is the information transfer phase in which the probe is successful and the call is in the process of transferring information. However, it is possible to incorporate the other phases into the compartmental model to be developed later in this section. The compartmental modelling requires the selection of appropriate compartments. The network is treated as a "black box", and the compartments are chosen in such a way that each compartment contains the calls that are in a particular state.

As in Garzia *et al.* (1989), we define the following compartmental variables for class i , $i = 1, 2, \dots, K$, at time t :

- $P_i(t)$: average number of calls probing the direct paths.
- $P_i^{1,j}(t)$: average number of calls probing the 1 st hop of the j th alternate path, where $j = 1, 2, \dots, r_i$.

- $P_i^{2,j}(t)$: average number of calls probing the 2nd hop of the j th alternate path, where $j = 1, 2, \dots, r_i$.
- $C_i^1(t)$: average number of established calls using direct paths.
- $C_i^2(t)$: average number of established calls using alternate paths.

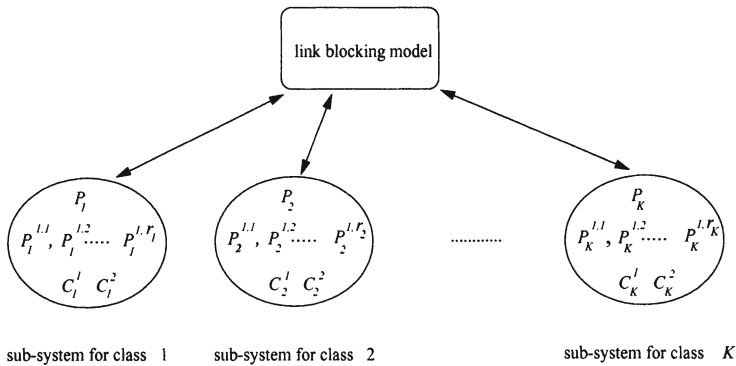


Figure 2 Compartmental sub-systems.

The compartments belonging to a given service class collectively form a compartmental sub-system as shown in Figure 2. In deriving the state equations, a sub-system is treated independently of others assuming that the link blocking probability of the corresponding service class is known. Moreover, depending on the bandwidth sharing policies (to be discussed later), the blocking probability of one service class may depend on the link offered load from the other classes. We account for the possible interactions among the compartmental sub-systems through link blocking as shown in Figure 2.

Difference Equations

Figure 3 illustrates the flow among the compartments of service class i . In deriving the state equations, the system is assumed to be in equilibrium during the interval $(t, t + \tau]$ (different equilibriums for different values of t), and the steady-state value of a variable at time $t + \tau$ is computed using its value at time t . By inspecting Figure 3, we can obtain the difference equations 5 through 10 relating the compartments of class i calls.

$$P_i(t + \tau) = \lambda_i \tau \quad (5)$$

$$C_i^1(t + \tau) = C_i^1(t) - \tau \mu_i C_i^1(t) + p_i(t) P_i(t) \quad (6)$$

$$P_i^{1,1}(t + \tau) = \begin{cases} q_i(t) P_i(t) & , r_i \geq 1 \\ 0 & , \text{otherwise} \end{cases} \quad (7)$$

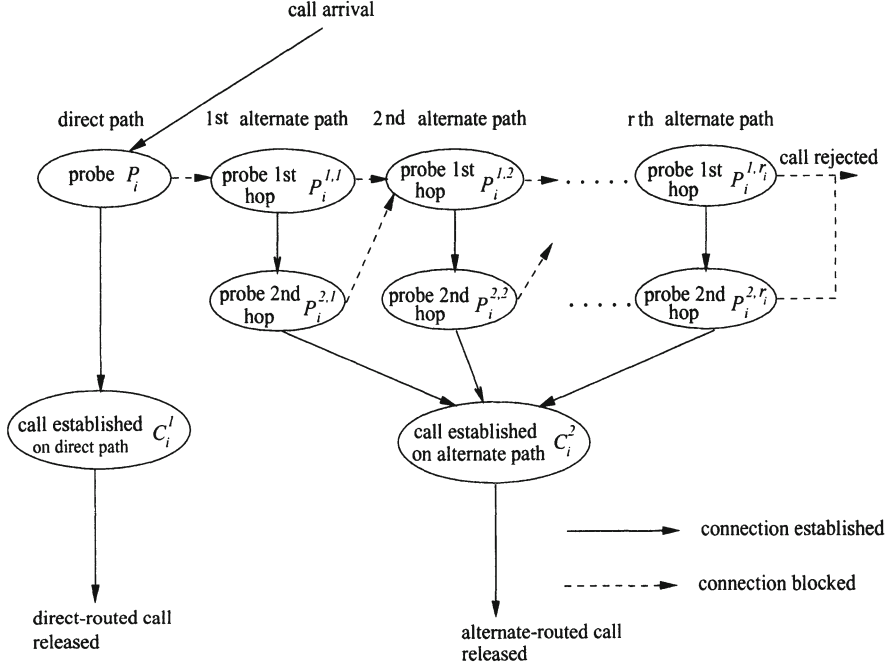


Figure 3 Flow diagram for service class i .

$$P_i^{1,j}(t + \tau) = \begin{cases} q_i(t) (P_i^{1,j-1}(t) + P_i^{2,j-1}(t)) & , r_i \geq 2, j = 2, 3, \dots, r_i \\ 0 & , \text{otherwise} \end{cases} \quad (8)$$

$$P_i^{2,j}(t + \tau) = \begin{cases} p_i(t) P_i^{1,j}(t) & , r_i \geq 1, j = 1, 2, \dots, r_i \\ 0 & , \text{otherwise} \end{cases} \quad (9)$$

$$C_i^2(t + \tau) = C_i^2(t) - \tau \mu_i C_i^2(t) + p_i(t) \sum_{j=1}^{r_i} P_i^{2,j}(t) \quad (10)$$

The above difference equations involve the link blocking probability $q_i(t)$, which depends on the link offered load $\rho_i(t)$. The latter is the ratio between the arrival and departure rates of the connections over a link. At time t , the arrival rate $L_i^a(t)$ of connections to any link from service class i can be computed as follows: each of the calls in the states $P_i(t)$, $P_i^{1,j}(t)$ and $P_i^{2,j}(t)$, where $j = 1, 2, \dots, r_i$, attempts to capture b_i BBU's on both links belonging

to a hop (since the call is bi-directional) in time τ . Since the total number of links in a fully connected network with N nodes is $N(N-1)$, the arrival rate is obtained as:

$$L_i^a(t) = \frac{2}{N(N-1)\tau} (P_i(t) + \sum_{j=1}^{r_i} P_i^{1,j}(t) + \sum_{j=1}^{r_i} P_i^{2,j}(t)). \quad (11)$$

In order to find the departure rate $L_i^d(t)$ of class i connections from a link, we note that the calls in state $P_i^{2,j}(t)$ will release the bi-directional connections established on the 1st hop of the j th alternate paths in time τ . Further, calls in state $C_i^1(t)$ and $C_i^2(t)$ will release one and two, respectively, bi-directional connections in time $\frac{1}{\mu_i}$ (time required to tear down a connection is neglected). The average departure rate is obtained as:

$$L_i^d(t) = \frac{\sum_{j=1}^{r_i} \frac{P_i^{2,j}(t)}{\tau} + \mu_i C_i^1(t) + 2\mu_i C_i^2(t)}{\sum_{j=1}^{r_i} P_i^{2,j}(t) + C_i^1(t) + 2C_i^2(t)}. \quad (12)$$

The link offered load from service class i is given by $\rho_i(t) = \frac{L_i^a(t)}{L_i^d(t)}$.

The equations 5 through 10 can be used to compute instantaneous values of the state variables at any given time t . Steady-state values of the variables can also be obtained by iterating these equations for sufficiently long time. Now all what we need is the blocking probability for each class for solving the state equations. The following section describes the procedures for computing the blocking probabilities.

Link Blocking Probability

We model the link as a multi-service loss system (Ross, 1995) in which an arriving connection request is either admitted or blocked. An admitted connection remains in the system for the duration of its holding time. The blocked connection is assumed to be lost. The bandwidth sharing policy dictates how link bandwidth is shared among different services. The blocking probability depends on the bandwidth sharing policy used. In this paper, we consider two popular sharing policies, namely *complete sharing* (CS) and *complete partitioning* (CP). During the iteration of the state equations, the blocking probabilities are recomputed at each step. Therefore, these quantities must be calculated using computationally efficient procedures to minimize the time required to carry out iterations. We now outline such procedures for CS and CP policies.

The connection requests from service class i arrive with traffic intensity ρ_i according to a Poisson process. Let us denote the state of the system

by $\mathbf{n} = (n_1, n_2, \dots, n_K)$, where n_i is the number of connections from class i . Under CS policy, a class i connection request is accepted on the link in state \mathbf{n} only if $\sum_{i=1}^K b_i n_i + b_i \leq C$. Using the recursive algorithm developed independently by Kaufman (1981) and Roberts (1981), the probability P_n that exactly n BBU's are occupied on the link can be computed as follows:

$$n P_n = \sum_{i=1}^K \rho_i b_i P_{n-b_i}, \quad n = 1, 2, \dots, C, \quad (13)$$

where $\sum_{n=0}^C P_n = 1$, and $P_n = 0$ if $n < 0$. The steady state blocking probability for class i connections is given by:

$$q_i = \sum_{n=C-b_i+1}^C P_n, \quad i = 1, 2, \dots, K. \quad (14)$$

Under CP policy, link bandwidth is divided into K partitions with capacities (C_1, C_2, \dots, C_K) where partition C_i is dedicated only to class i . So, in state \mathbf{n} , a new class i connection request is accepted only if $(n_i + 1)b_i \leq C_i$. Since there is no overlap among the partitions, the blocking probability for class i connections can be computed independently of other classes using the Erlang-B formula:

$$q_i = B(M_i, \rho_i) = \frac{\frac{\rho_i^{M_i}}{M_i!}}{\sum_{n=0}^{M_i} \frac{\rho_i^n}{n!}}, \quad (15)$$

where M_i is the largest integer less than or equal to $\frac{C_i}{b_i}$. $B(M_i, \rho_i)$ can be computed fast using the following recursive formula:

$$\frac{1}{B(n, \rho_i)} = 1 + \frac{n}{\rho_i} \frac{1}{B(n-1, \rho_i)}, \quad i = 1, 2, \dots, M_i, \quad (16)$$

with $B(0, \rho_i) = 1$.

5 A NUMERICAL EXAMPLE

In this section, we use the compartmental model to evaluate the steady-state performance of a fully connected 10-node network. The network supports two service classes ($K = 2$), namely *class 1* and *class 2* services. Each link has a bandwidth of 100 BBU's. A class 1 call requires 1 BBU and has a mean holding time of 1 minute, whereas a class 2 call requires 3 BBU's and has a mean holding time of 3 minutes. We further assume that 75% of the arriving calls are from class 1 and the remaining 25% are from class 2. Let's denote the aggregate call arrival rate from both class 1 and 2 to the network by λ^a . Then, $\lambda_1 = 0.75\lambda^a$ and $\lambda_2 = 0.25\lambda^a$. These parameters are selected in such

a way that each service class offers equal load to the network, i.e., $\frac{\lambda_1}{\mu_1} = \frac{\lambda_2}{\mu_2}$. We set the number of alternate paths to 6 for both classes ($r_1 = r_2 = 6$), and the network load is changed by varying λ^a . The time step τ used for this study is 0.1 second. The system is brought to the steady-state after iterating the state equations for 5000 seconds (requires 50 000 iterations). We use the average number of calls in the network as the performance metric. For class i calls, the value of this metric is obtained as a summation of C_i^1 and C_i^2 .

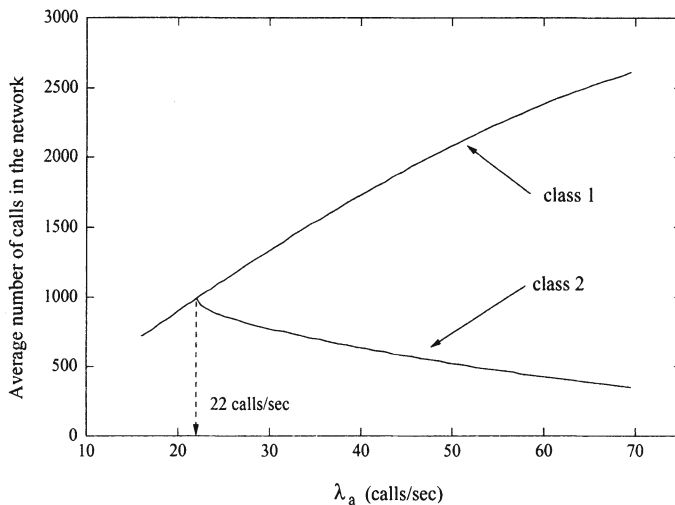


Figure 4 Variation of average number of calls in the network with λ^a under CS policy

Figure 4 shows the impact of λ^a on the number of calls in the network under CS policy. As λ^a increases, the number of calls from both classes increases steadily until λ^a reaches approximately 22 calls/s. As λ^a increases further, the number of class 1 calls continues to increase whereas that of class 2 calls decreases. We observe that, under heavy load ($\lambda^a > 22$ calls/s), the link bandwidth is monopolized by class 1 calls, i.e., the network is unfair to class 2. This observation supports the well known fact that CS policy favours services with smaller bandwidth requirements.

Class 2 calls can be protected using other bandwidth sharing policies. For example, let us consider CP policy. Since there is no interference between the service classes, competition for link bandwidth is avoided. Figure 5 shows the variation of the number of calls in the network with λ^a under CP policy. Two partitioning cases are considered. In the first case, link bandwidth is equally divided between the two classes, and in the second case, 75% of the link bandwidth is allocated for class 2 and the rest is allocated for class 1. Note that, in the second case, a partition for a class is proportional to the bandwidth requirement of the class. Unlike CS policy, CP policy keeps the number of class 2 calls steady at higher load. However, from Figure 4 and 5, we clearly see that the network performance with respect to class 2 service

is improved at the cost of blocking more class 1 calls. Note also that, under CP policy, the number of calls is sensitive to the partition setting. With the first setting, the number of class 1 calls is much higher than that of class 2, whereas the second setting results in equal number of calls from each class throughout the entire loading region.

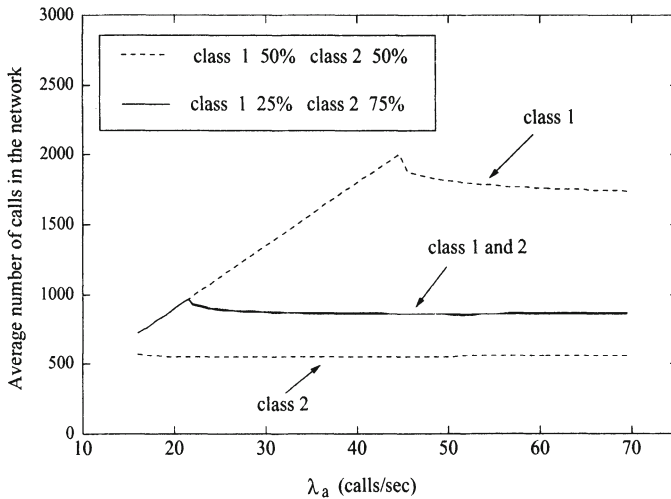


Figure 5 Variation of average number of calls in the network with λ^a under CP policy.

We will now have a closer look at the behaviour of the network when λ^a approaches 22 calls/sec (in the increasing direction) under CS policy (see Figure 4). For the purpose of this discussion, let us call this rate the critical rate. Figure 6 illustrates the impact of λ^a on the link blocking probability. Observe that, below the critical rate, both classes experience very small blocking. But when λ^a slightly exceeds the critical rate, the link blocking probability suddenly becomes very high for both classes, with class 2 calls experiencing higher blocking than class 1 calls. This phenomena can be explained as follows: as λ^a approaches the critical rate, the link blocking becomes significant and hence calls probing the direct path experience more blocking. Consequently, the number of alternate-routed calls begins to increase. The increasing alternate-routed calls offer more load to the links (because they use two hops) and therefore lead to a further increase in the link blocking. This “domino” effect causes the drastic increase in the link blocking probability and the number of alternate-routed calls. Since an alternate-routed call uses twice the amount of resources as a direct-routed call and the amount of network resources is fixed, the domino effect causes the total number of calls to drop quickly as λ^a exceeds the critical rate. This is evident from the slope of the curve for class 2 in Figure 4 in the vicinity of the critical rate. Similar observation has been made by Akinpelu (1984) in telephone networks.

For the selection of the network parameters used in this example, class 1

calls are not affected by the domino effect. This is because under CS policy, class 1 service gets the extra bandwidth required to accommodate its increasing alternate-routed calls by further depriving the class 2 service of the link bandwidth. Under CP policy, class 1 service also exhibits a sudden reduction in the number of calls due to domino effect (see Figure 5). In this case, however, the value of λ^a at which the domino effect begins is different from that under CS policy, and depends on how the link is partitioned. Class 1 service now obtains the additional bandwidth required to accommodate its increasing alternate-routed calls from its own partition as the links switch to a higher blocking state. So, the increase in the alternate-routed calls causes a sudden drop in the total number of class 1 calls in the network.

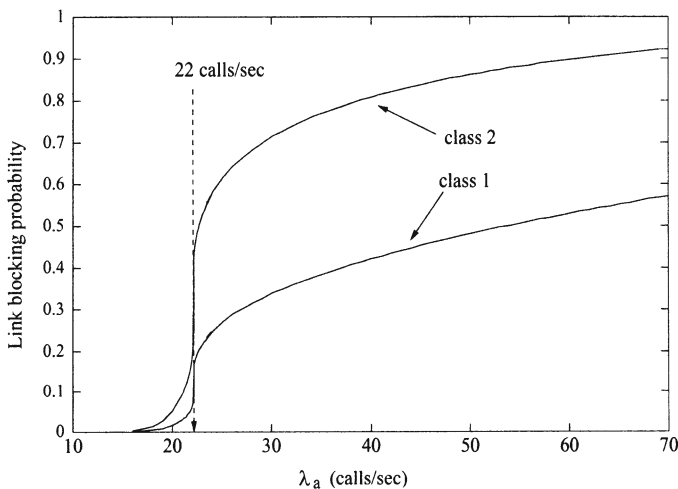


Figure 6 Variation of link blocking probability with λ^a under CS policy.

6 ACCURACY OF THE COMPARTMENTAL MODEL

The compartmental model aggregates the system description into just a few state variables and is based on the solution of the classical single-link blocking model. These features reduce the complexity of solving the state equations. Due to somewhat oversimplifying assumptions, we cannot expect the compartmental models to yield very accurate results. In this section, we use call level discrete-event simulations to assess the the results obtained using the compartmental model.

For the simulations, we consider an alternate routing scheme (responsible for selecting alternate paths) known as the *least busy alternate routing* in which the alternate path with the highest residual capacity is chosen first (see Ross, 1995 for details). We consider only CS policy. Figure 7(a) and (b) shows the variations of link blocking probability (averaged over all links) and the number of calls respectively with the load. In the region where $\lambda^a \leq 19.5$, the compartmental model yields results that are in good agreement with those of

simulations. As the load increases the simulation results quickly deviates from the analytical results. However, the critical aspects of the network behaviour remain the same. For instance, according to the simulation results, the number of class 2 calls begins to drop when λ^a approaches 19.5 calls/sec (in the increasing direction) and the onset of this event is associated with a drastic increase in the link blocking. The compartmental model also predicts a similar trend. In this case, the number of class 2 calls begins to drop when λ^a slightly exceeds 22 calls/sec (note also the drastic increase in the link blocking probability in Figure 6).

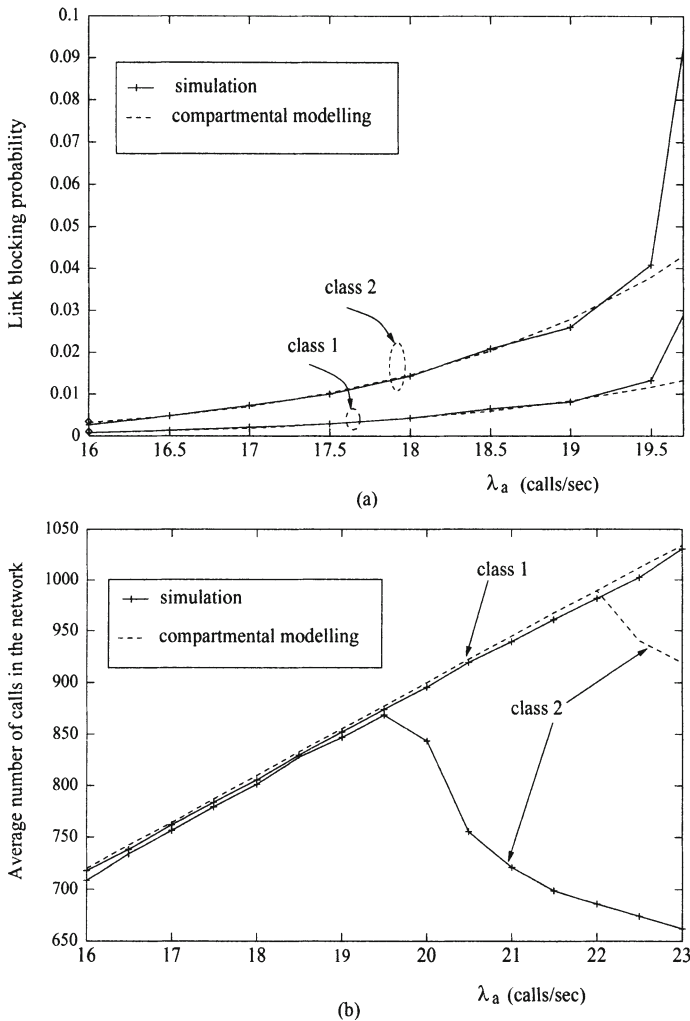


Figure 7 Comparison of compartmental modelling with simulations with respect to (a) link blocking probability (b) average number of calls.

In compartmental modelling, we do not consider a particular routing

scheme. Instead we assume that the routing scheme operates in such a way that the load is uniformly spread across the network. This condition may be difficult to achieve by a real routing scheme such as the one used for our simulations. We have also carried out additional simulation with different parameters, e.g., network size, bandwidth sharing policy, etc. The main observation is that the compartmental model predicts the critical aspects of the network behavior.

7 CONCLUSIONS

The performance of communication networks can be evaluated through simulation. It is easy to develop simulation models that capture the fine aspects of network elements. A major concern with this approach is the long simulation time. On the other hand, it is very difficult to develop analytical models which accurately represent all the characteristics of a network. Furthermore, many analytical models developed are suitable only for steady-state analysis.

Compartmental modelling technique yields a good compromise between accuracy and complexity of analytical approach, and can be used for both steady and transient analysis of complex networks. This technique has been applied to connection-oriented symmetric networks carrying a single service class. We have shown how this technique can be easily adapted to multi-service environment, and demonstrated its application to the steady-state performance analysis of a 10-node network supporting two service classes. Comparison with simulation results indicates that, the compartmental models can be useful for qualitatively investigating the critical aspects of symmetric networks.

8 REFERENCES

- Akinpelu, J. M. (1984) The Overload Performance of Engineered Networks with Non-hierarchical Routing, *AT & T Bell Labs. Tech. J.*, Vol. **63**, No. **7**, 1261-81.
- de Prycker, M. (1991) Asynchronous Transfer Mode: A Solution for Broadband ISDN, Ellis Horwood Ltd., New York.
- Garzia, M. R. and Lockhart, C. M. (1989) Nonhierarchical Communications Networks: An Application of Compartmental Modeling, *IEEE Transactions on Communications*, Vol. **37**, No. **6**, 555-64.
- Godfrey, K. (1983) Compartmental Models and Their Application, Academic Press, London.
- Kaufman, J. S. (1981) Blocking in a shared resource environment, *IEEE Transactions on Communications*, Vol. **29**, No. **10**, 1474-81.

- Lee, W. C., Hluchyi, M. G., and Humblet, P. A. (1995) Routing Subject to Quality of Service Constraints in Integrated Communication Networks, *IEEE Network Magazine*, Vol. 9, No. 4, 46-55.
- Roberts, J. W. (1981) A service system with heterogeneous user requirements, *Performance of Data Communications Systems and their Applications*, 423-31, North-Holland.
- Ross, K. (1995) Multiservice Loss Models for Broadband Telecommunication Networks, Springer, New York.

9 BIOGRAPHY

M. Sivabalan received the B.Sc. degree in Electrical Engineering from the University of Peradeniya, Sri Lanka, and the M.Sc. degree in the area of Cryptography from the Department of Electrical and Computer Engineering, Queen's University, Kingston, Ontario, Canada, where he is now a Ph.D. candidate. His current research area is routing in high speed networks.

H. T. Mouftah is a Professor in the Department of Electrical and Computer Engineering, Queen's University, Kingston, Ontario, Canada since 1979. Prior to that, he worked with the Data Systems Planning Department at Nortel Technology (then BNR) of Ottawa for three years. He has also spent his sabbatical years 1986/87 and 1993/94 with Nortel Technology working in the area of high speed integrated networks, routing, and traffic management. Since 1989, he is a Principal Investigator for the Telecommunications Research Institute of Ontario (TRIO), a government Centre of Excellence in Communications responsible for a project on Broadband Packet Switching Networks.

Dr. Mouftah is the recipient of the 1989 Engineering Medal for Research and Development of the Professional Engineers of Ontario (PEO). He is the joint holder of an Honorable Mention for the Frederick W. Ellersick Prize paper award for the best paper in the Communications Magazine in 1993. He is also the joint holder of the Outstanding Paper Award for a paper presented at the IEEE 14th International Symposium on Multiple-Valued Logic (ISMVL'84). He is the recipient of the IEEE Canada (Region 7) Outstanding Service Award (1995). Dr. Mouftah is a fellow of the IEEE (1990), and Member of the PEO and the Canadian Association of University Teachers.

G. Takahara received the B.A. degree in Mathematics from the University of British Columbia, Vancouver, B.C., and the M.S. and Ph.D. degrees in Statistics from Carnegie Mellon University. Since 1994, he has been an Assistant Professor in the Department of Mathematics and Statistics at Queen's University in Kingston, Ontario, Canada. His research interests are in applied probability modelling in communications networks, loss networks, retrials, circuit switching, and queueing theory. He is a member of the IEEE Communications Society.

Performance Evaluations of Partial Order Connections

M. Fournier, C. Chassot, M. Diaz, A. Lozes

LAAS-CNRS

*7 avenue du colonel Roche 31077 Toulouse Cédex, France,
fournier, chassot, diaz, alozes@laas.fr*

Abstract

Technological improvements in Computer and Telecommunication areas lead to the emergence of new distributed multimedia and co-operative applications, whose features and requirements are both multiple and diversified. In order to tackle their needs, two major approaches are currently developed, one arguing for an improvement of the application software, the other one promoting a sophisticated network support. Following this last approach, a new concept of connection, *the partial order connection* (POC), has been proposed and formalized. This paper provides simulation results of the POC concept obtained using the OPNET software tool. Particularly, it is shown how a partial order / partial reliability management at the transport layer impacts on end to end transit delay and on the memory used at both sending and receiving transport entities.

Keywords

Multimedia Application, Transport Service, Transport Protocol, Partial Order Connection, Performance Evaluation, OPNET

1 INTRODUCTION

During the last decade, technological improvements in Computer and Telecommunication areas led to the emergence of new distributed *multimedia* and *co-operative* applications, including transmission and processing of all data types (text, fixed or animated image, voice, video sequences, etc.). Among the requirements of such applications, the following ones are now of importance:

- high speed transmission links are obviously needed;
- temporal constraints have to be enforced for interactive communications such as visioconferencing systems;
- multimedia synchronization is required: exchanged data have to be orchestrated the ones with the others;
- finally and at the opposite of file data transfers, distributed multimedia applications tolerate an imperfect transfer of the data they involved, without being unacceptable for users.

A full analysis of distributed multimedia applications requirements may be found in [23].

Current solutions and their limits: Recently, many studies have been performed to design light weight transport protocols (such as NETBLT [6], VMTP [4] or XTP [17]) more suited to support multimedia data transfers than TCP or TP4 protocols; however, these proposals do not provide a sufficient solution with respect to multimedia requirements. Usually, multimedia synchronization issues (both spatial and temporal ones) are not addressed and temporal constraints management mechanisms are implementation dependent. A new generation of high speed, multimedia protocols has to be designed on top of high speed links using ISDN, Internet protocols or ATM. Presently, several communication architectures are envisaged within different projects which may be classified as follows.

The "Application Aware Networking" approach: Up to now, two major architectural approaches have been proposed to design new distributed communication systems; both extend the *Application Layer Framing* (ALF) concept introduced in [7] that argues the interest of Application Data Units (ADU) preservation at the communication system level:

- the first approach assumes that the network software has to be as simple as possible without providing any Quality of Service (QoS) guarantee. The application is supposed to be the most appropriate to adapt to network fluctuations by integrating in its software the most adequate transport mechanisms. This *Network Aware Application* (NAA) approach is mainly supported by the Internet community, and is currently pursued within the Esprit project HIPPARCH. Recent investigations and results are given in [5] [14] [13].
- at the opposite, the second approach suggests to increase the networking system complexity to handle new requirements defined at the applicative level. The major interest of this approach is to make the user software simpler at the cost of a more sophisticated network support. This *Application Aware Networking* (AAN) approach has been or is currently pursued within several projects: OSI 95 [8], RACE CIO [16], BERKOM [9], QoS-A [3], TENET [15] and CESAME [11].

This paper, which follows the AAN approach, will evaluate the performance of a new kind of connection, the *partial order connection* (POC), introduced in [1] and [10]. A POC defines a new transport QoS related to 'order' and 'reliability' parameters; when implementing this new concept, a transport protocol allows bandwidth, memory and transit delay savings. The major part of this paper aims at showing (in a simulation environment) the interest of the POC concept for two parameters: the number of used buffers and the applicative transit delay. This paper is divided into five sections. Basic principles of the POC concept are first recalled in section 2. Tested partial order transport protocol is composed of different mechanisms which are briefly exposed in section 3. In section 4, the communication model (including network support) is presented

first. Simulation context is then exposed: the selected parameters are given and their different values during simulations are detailed; simulation results are finally given and analyzed. Conclusions are stated in section 5.

2 PARTIAL ORDER CONNECTION: CONCEPT AND ANALYSIS

At the present time, commonly used transport protocols are either based on the connection-oriented (CO) mode or on the connectionless (CL) one:

- on one hand, CO protocols (such as TCP) provide their users with full reliability and total order, at the cost of increased delay and reduced throughput;
- on the other hand, CL protocols (such as UDP) introduce no increase in transit delay or reduction in throughput but provide neither order nor reliability guarantee.

This classification shows a design gap between CO and CL protocols that suggests a conceptual extension of the classical connection concept, using *order* and *reliability* parameters. This extension has led to the *partial order connection*, for which CO and CL protocols are two special cases. POC has been introduced in [1] and [10]. The basic principles of the POC concept and its suitability for multimedia applications are presented in 2.1. In order to quantify and then compare the complexity of different partial order services, quantification metrics have also been introduced in [1] and [10]; in order to classify the different protocols whose benefits are evaluated in section 4, a brief overview of the used metric is presented in 2.2.

2.1 POC Concept

A conceptual extension of the connection concept: A POC is an end-to-end connection that allows its users to define and use for transferring data any partial order/reliability service from no order/no reliability (typically a UDP-like service) to total (sequential) order/total reliability (typically a TCP-like service). In a POC, 'order' and 'reliability' appear as two specific QoS parameters specified by the service user in the connection set-up request. Once known by the sending entity, 'order' and 'reliability' are translated into protocol parameters used to start-up associated protocol mechanisms. In a POC, service data units (SDUs) can be delivered to the receiving user in an order that is different from the sending order: the acceptable difference between the submission sequence and the different but acceptable delivery sequences precisely results from the definition of the selected partial order.

A concept suitable for multimedia applications: Up to now, a few models have been proposed in order to provide a formal representation of temporal constraints in distributed systems; most of them are based on Petri nets [22] [19] [2]. One of this model, the Timed Stream Petri Net (TSPN) model [12] [21], provides a formal description of multimedia synchronization scenarios in asynchronous distributed systems. In this model, a place is associated with a data presentation (a still image or a sound fragment for instance); logical

dependency relationships existing either within a flow or between the different flows of the application are modeled using transitions of the net. The Petri net deduced from a TSPN model provides a logical representation of temporal synchronization constraints, which is a partial order.

Consider for instance the *multimedia* object pictured on the left part of Figure 1; it is composed of different *monomedia* objects (a logo, two fixed images and two video sequences numbered from 1 to 99) which will have to be displayed at the screen of distant user; the picture gives the expected object display at the receiving side. The right part of the figure provides the partial order, say P , deduced from the application defined TSPN model, that illustrates (among other) that logo 1 and image 2 may be distributed at the receiving side independently the one of the other as they are in parallel both in the display and in the model; logical synchronization constraints within the two video sequences are expressed by the intermediate transitions t_i .

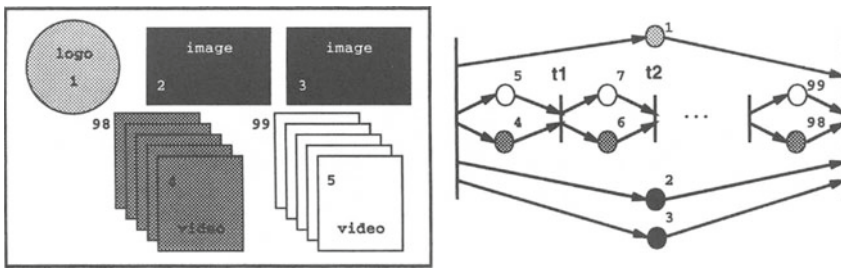


Figure 1 Partial order deduced from a TSPN model of a multimedia object

Provided with a P partial order transport connection, the receiving user may then be delivered monomedia objects (typically SDU) in any sequence consistent with both spatial and temporal synchronization constraints. Moreover, as it is shown through different theoretical examples in [1], these different delivery sequences lead to transfer speed-up and save resources at both sending and receiving sides. Simulations given in section 4 will confirm these first results.

2.2 Quantification metric

A simple 'reliability' measure consists (for instance) in the maximum number of losses the protocol may tolerate without recovering. Divided by the number of sent objects during the observation period, this measure is still between 0 and 1. Such an 'order' measure is not obvious and several metrics have been proposed in [1] and [10]. The one we select to keep a representative subset of all protocols (according to order and reliability) is the following one [1]:

$$m(Z) = \frac{\ln(e(Z))}{\ln(\text{card}(Z)!)}; \text{ with } Z: \text{ number of objects / period.}$$

3 PROTOCOL DESIGN

Performance evaluations exposed in section 4 have been realized using a protocol based on a selective positive acknowledgment mechanism. This protocol has been initially modeled using the formal description technic Estelle [18]; the Estelle code may be found in [1].

At the sending side, PDUs are numbered and then buffered in the event of future retransmissions. A timer is associated to each sent PDU. A flow control is also ensured by means of a sending window.

At the receiving side, any received PDU is acknowledged and its "deliverability" with respect to order and reliability QoS parameters is checked; three cases are considered:

- The PDU is deliverable; it is then delivered to the user.
- The PDU is not deliverable (order and reliability constraints are not respected); it is acknowledged and then buffered until these constraints are verified.
- The PDU has already been received (it is a duplicate); it is then rejected.

In addition to order and reliability parameters, two other parameters are managed by the protocol: the sending window length and the retransmission timer. In order to optimize retransmissions management, the value of the timer has to be updated according to the Round Trip Time (RTT) value. When the underlying network only provides a "best effort" service (typically, the present Internet), it is not possible to have a fixed timer value (RTT variations are not predictable and may be very important); however, in order to make our simulations simpler, we fixed the retransmission timer value to $1.1 \cdot \text{RTT}$. Sending window size has been fixed to 10 PDUs. Note that these two parameters can be modified for other simulations.

4 PERFORMANCE EVALUATIONS

Protocol design and performance evaluations have been made using the OPNET tool [20]. OPNET allows one to design a hierarchical and structured communication architecture (an OPNET model) composed of three different design levels, network, node and process levels, representing for instance an ATM network, a gateway (or a workstation) and its behaviour respectively. In the following, our model is first presented and its parameters are detailed (4.1). The simulation context, i.e. parameters and their values, is detailed in (4.2). Their impact on both sending/receiving buffers and applicative transit delay is exposed and analyzed.

4.1 Model description

At the network level, the model is composed of three parts, the sender, the receiver and the network (see figure 2). The sender is composed of a data

generator and the sending POC entity. The receiver is composed of the receiving POC entity and a data consumer.

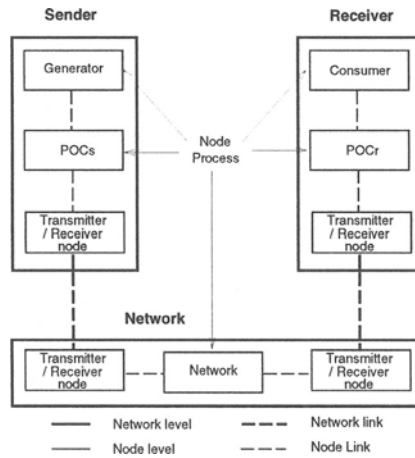


Figure 2 Partial order Model

Sender parameters: The sender parameters are the packet generation date and the sending period duration. Time origin is given by the first packet sending date.

Receiver parameters: The receiver parameters are 'order' and 'reliability' QoS. Reliability is expressed by a real between 0 and 1, 0 meaning unreliable service and 1 reliable service. Order is expressed by an integer list representing the immediate predecessors of each object (see example given in figure 3). For instance, object 1 of the period (composed of 4 objects) is preceded by 2 objects (predecessor number = 2): -1 et 0 (predecessor list = -1 et 0).

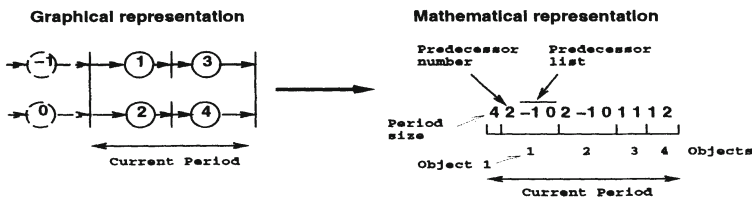


Figure 3 Order representation

Network parameters: Two parameters are considered for the network: transit delay and loss rate.

- Transit delay is expressed in milliseconds and represents the time between one outgoing PDU (at the sending side) and one incoming POC PDU (at the receiving side).

- Loss rate is expressed by a real number between 0 and 100. Losses are produced by a random generator; generated numbers are extracted from an uniform distribution between 0 and a maximum value set up by the user.

The protocol allows simulation of different order and reliability with variable transit delay and/or loss rate values. Note that simulations are reproducible because the initial value of the OPNET random generator (used to produce random losses) is a simulation parameter.

4.2 Simulation context

The simulation context defines parameters values that have been used in our simulations. The two major parameters are the order and reliability QoS parameters. Other parameters characterize both the network (loss rate, transit delay) and the transport protocol (sending window size, retransmission timer value). Simulation duration is the last defined parameter.

(a) QoS

Partial order: Ten partial orders have been tested. All are serial/parallel Petri nets compositions of eight objects. Their representation on the order axis given on figure 5 results from the metric m described in section 2.2. The ten partial orders have been chosen to be equitably distributed on the order axis. Details of these orders are given on figure 4.

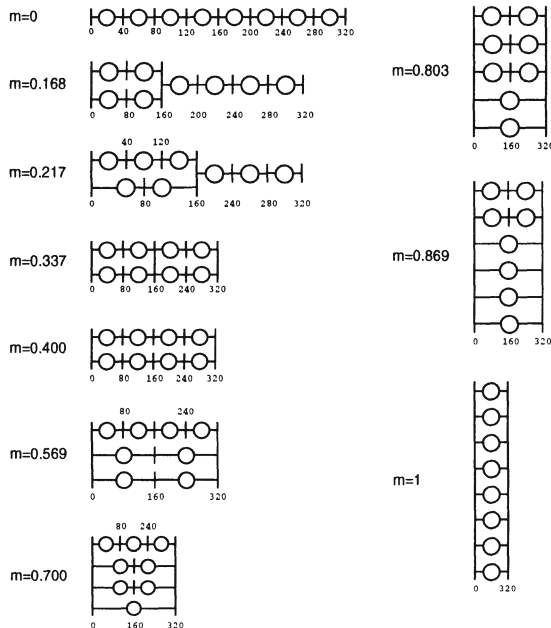


Figure 4 Set of serial parallel compositions

Partial reliability: Reliability is represented between 0 and 1, which respectively represents unreliable and reliable service. For instance, a 0.9 reliability indicates that 10% of user data may be lost by the transport service provider. We chose reliability between 0.5 and 1, this variation being sufficient enough to show reliability impact on both buffers and transit delay. Set of tested values is 1, 0.99, 0.95, 0.9, 0.75 and 0.5.

QoS representation: As it has been described in section 2.2, it is possible to classify any protocol in a two axis system ('order', 'reliability') such as the one given on figure 5. Protocols which will be evaluated in the following are localized in this figure by means of character "o".

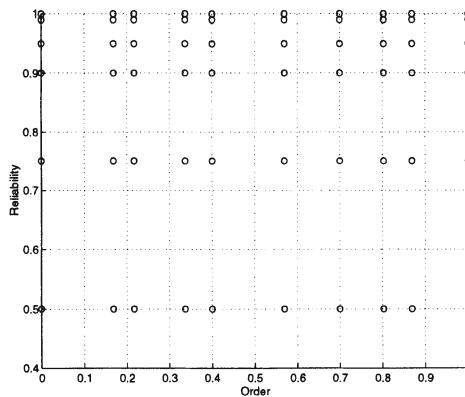


Figure 5 Protocols set

(b) Simulation parameters

Transit delay: Transit delay parameter allows one to simulate different networks. Our goal being first to measure the impact of the QoS parameters, we fixed the transit delay value to 50ms, representing a RTT of 100ms.

Loss rate: Order and reliability impact appears only when losses occur. It is then necessary to introduce them in the network; four loss rates have been tested: 0, 5, 10 and 25%. Even if a network with a 0% loss rate is not realistic, associated simulation results will be given here as a reference level.

Retransmission timer and sending window size: Retransmission timer has been fixed to $1.1 * RTT$. The sending window size is 10 PDUs, a size which appeared during simulations being sufficient to preserve data flow continuity.

(c) Simulation duration

An important point to determine before experiment start up is the optimal simulation duration. Indeed, simulations have to be made in a stationary rate, not during a transitory one. On figure 6 where both buffers and transit delay evolution are presented, we see that the transitory phenomena is relatively short compared to the 1000s simulation duration. Illustrated curves have been obtained with maximal order/reliability constraints (that is total order/reliability) and

a 10 % loss rate. We assumed for all curves that the same simulation duration was sufficient to reach 'permanent rate' results.

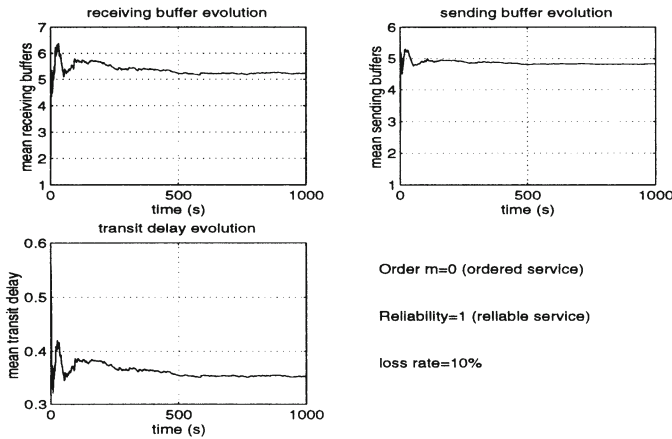


Figure 6 Transitory and permanent rates

4.3 Partial Order Connection evaluation

(a) Order impact on sending/receiving buffers

In this subsection, evaluation of mean sending/receiving used buffers is computed for each of the ten tested partial orders, depending on four network loss rates (0, 5, 10 and 25 %) and three particular reliability values:

- 0, which represents a totally reliable service.
- 0.95, which represents an unreliable service with a 5% acceptable loss rate.
- 0.5, which represents an unreliable service with a 50% acceptable loss rate.

Sending (respectively receiving) buffers are given on figure 7 (respectively on figure 8). Only 'permanent rate' results are provided on these figures. Each figure is composed of four sub-figures (one for each network loss rate), each of them representing three curves (one for each reliability).

Sending buffers (figure 7): As any received or declared lost PDU is acknowledged by the receiving POC entity (out of partial order received PDU are stored), note that for a given (reliability, loss rate) couple, order impact on used sending buffers is null (same used buffers number for each of the ten selected orders). However, it is to be noticed that used sending buffers number depends on the (reliability, loss rate) couple value. This last point will be detailed in subsection (c) ('Reliability impact on sending/receiving buffers').

Receiving buffers (figure 8): Order impact on receiving buffers appears to be more significant. Indeed, as any out of order received PDU is stored at the receiving side, the less the order constraints are, the less the receiving buffers are full. This point is clearly shown through the decreasing slope of the different curves. As far as this lastest point is concerned, one can also notice that slope

of the different curves also depends on the (reliability, loss rate) couple value. This point will be analyzed in subsection (c).

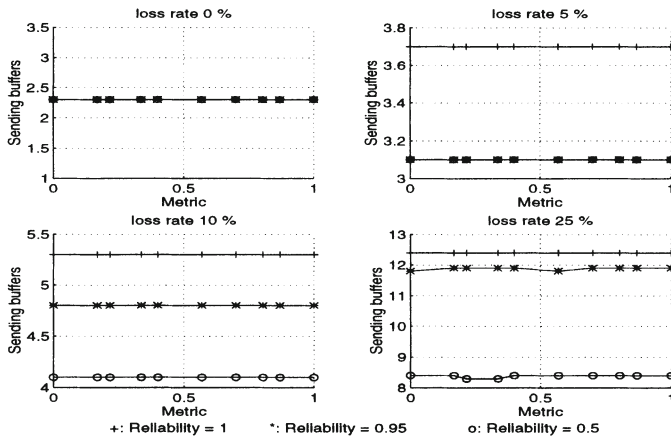


Figure 7 Sending buffers

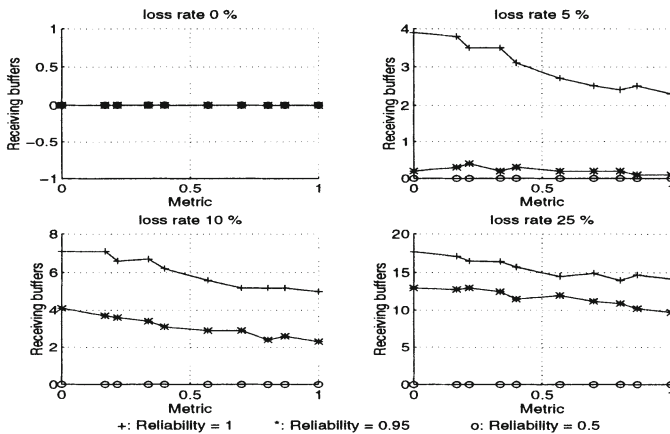


Figure 8 Receiving buffers

(b) Order impact on the transit delay

Order impact on the applicative transit delay is shown on figure 9. Except on the first curve (loss rate = 0%), the following conclusions may be stated:

- All curves are decreasing with order (that is when order constraints are decreasing). For instance, for the curve obtained for a 10% loss rate and a 0.95 reliability, a 20% saving on the transit delay can be observed for the partial order whose metric is 0.569 compared with the total order ($m = 0$). Instead of having to buffer an out of sequence (i.e. out of total order) PDU and wait for the arrival of the retransmitted PDU(s) (lost by the network

during their first transfer) that precede it in the total order, the receiving POC entity may deliver such a PDU as soon as it has been received provided its delivery matches the selected partial order.

- Another point has also to be noticed: order impact on the transit delay is less significant when the reliability parameter gets closer to or is greater than the network loss rate (see all the curves obtained for a QoS reliability or the one given for a 0.95 reliability and a 5% loss rate). Indeed, in these cases, lots of retransmissions are avoided (the selected reliability service still being ensured), making the previous point less significant.

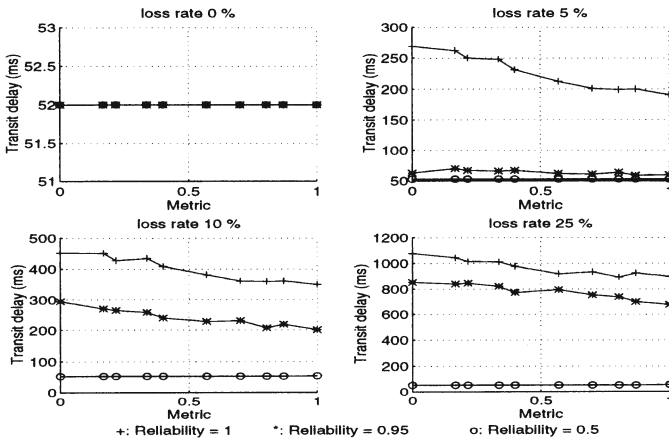


Figure 9 Transit delay evolution

(c) Reliability impact on sending/receiving buffers

In this subsection, reliability impact on used sending/receiving buffers is evaluated. The different loss rates which have been kept are the same as in the previous two subsections: 0, 5, 10 and 25%. Only three representative orders have been selected:

- 0, which represents total order (serial compositions only).
- 0.569, which represents an intermediate partial order between total order and null order (serial and parallel compositions).
- 1, which represents total disorder (parallel compositions only).

Sending (respectively receiving) buffers evolution is given on figure 10 (respectively on figure 11).

Sending buffers (figure 10): Partial reliability management allows the receiving POC entity to acknowledge lost PDUs (no retransmission will be performed) if these losses still match the selected reliability service; remember that a PDU is considered to have been lost by the network when a PDU that follows it in the partial order has been received. It then appears that when relaxing

the reliability constraints, the sending used buffer are freed sooner, which is expressed on curves of figure 10 by the decrease of the mean used sending buffer numbers. Two other points have also to be noticed:

- Reliability impact on sending buffers is significant only when the reliability parameter is greater than the network loss rate; else, the protocol has not to process retransmissions since all losses generated by the network do not damage the required reliability service. A clear illustration of this last point is given for a 5% loss rate. Reliability impact is null if the corresponding service is less than 0.95; beyond this value, buffers saving significantly appears (about 20% for 0.95 reliability service compared with a totally reliable service).
- One can also notice that less buffers are used for a totally ordered service compared with a partial order one. This observation results from the protocol loss declaration mechanism. Indeed, this mechanism is activated by the receiving POC entity when an undeliverable PDU (regard with the selected partial order) is received and all not yet received PDUs that precede it in the partial order may be declared lost still ensuring the required reliability service. In this case, these PDUs are declared lost and then acknowledged by the receiving POC entity. Total order having more order constraints than other tested orders, previous acknowledgments are generated earlier, making sending buffers to be freed at the earliest time.

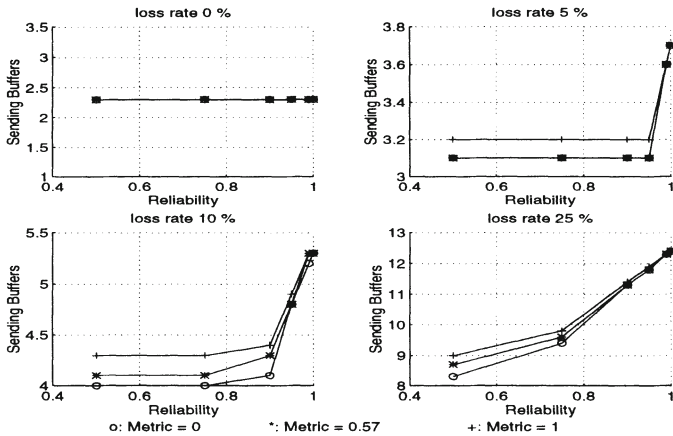


Figure 10 Sending buffers

Receiving buffers (figure 11): Analogous conclusions can be stated about reliability impact on receiving buffers. Particularly, it clearly appears that this impact becomes significant when losses generated by the network are greater than the user acceptable loss rate. For instance, if we look at the curves obtained for a 10% loss rate, about 50% of the used buffers are saved when a 0.95

reliability is provided compared with a total reliability (order=0.569); for a reliability lesser than 0.9, this gain no more appears.

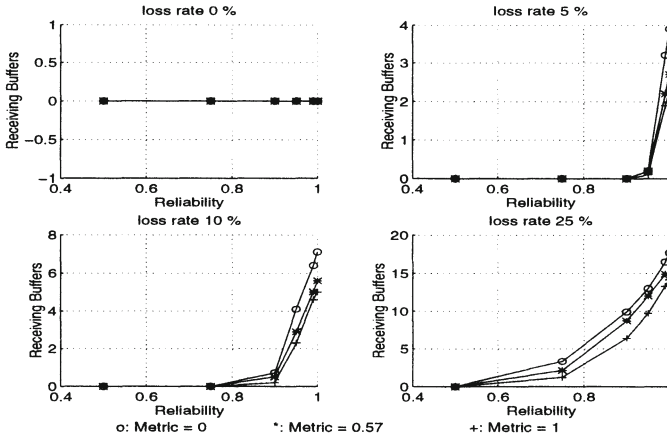


Figure 11 Receiving buffers

(d) Reliability impact on the transit delay

Reliability impact on the transit delay is shown on figure 12.

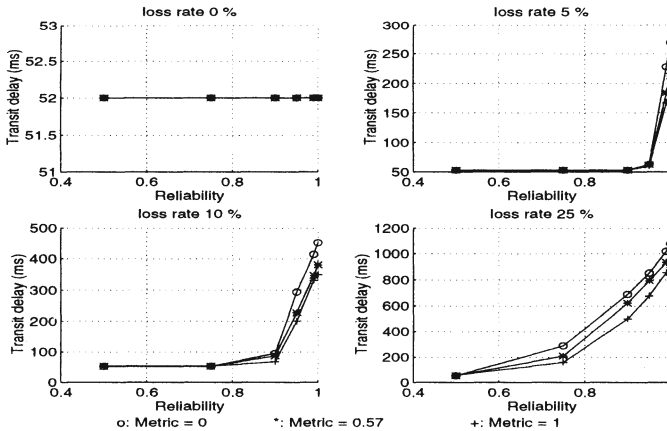


Figure 12 Transit delay

- For all curves, the lowest transit delay values are obtained when the network loss rate is lesser than the user acceptable loss rate; in this case, the protocol has not to process loss recovery and then does not increase the delay.
- Out of this particular context, benefits generated by a partial reliability management clearly appear. For instance, transit delay measure obtained with a 10% network loss rate is 40% lesser for a 0.95 reliability than for a total one (order=0.569). Such a gain (180 ms in the previous example)

allows service users to manage temporal synchronization constraints more easily, still ensuring reliability guarantees.

5 CONCLUSION

This paper has presented a few results performed to characterize a new concept of connection, the partial order connection, in a simulation environment (OPNET). The goal of this paper was to provide an evaluation of the benefits, related to transit delay and memory, a transport protocol implementing the POC concept generates. First, the basic principles of a partial order connection have been recalled; to classify and then compare the different partial order protocols tested in the paper, a metric, m , has also been detailed. Second, main mechanisms our POC protocol is constituted of have been presented to address the two major QoS parameters, 'order' and 'reliability'. The OPNET simulation environment has then been introduced and the communication model has been described. Simulation results have provided a first estimate of the benefits a POC may generate within an imperfect network environment, compared with a classical TCP-like connection. The following general conclusions may be stated.

Concerning order, it first appears that the less the order constraints are, the less buffers are needed and the lower applicative the transit delay is; however, note that the order parameter is entirely application dependent and cannot be modified, which is not exactly the case for the reliability (as users may modify it). When the network loss rate is lesser than the user required reliability, reliability management at the transport level is not useful, this QoS being ensured by the network itself. In the general case, that is when the network does not provide a sufficient enough reliability with respect to the application needs, very significant benefits have been shown when a partial reliability management is implemented at the transport level. Particularly, transit delay improvements when total reliability is not needed is extremely interesting for time sensitive or real time distributed applications, as the synchronization constraints of such applications are very difficult to enforce at the application layer.

REFERENCES

- [1] P.D. Amer, C. Chassot, C. Connolly, P. Conrad, and M. Diaz. (1994) Partial order transport service for multimedia and other applications. *IEEE/ACM Transactions on Networking*, vol.2, n°5.
- [2] B. Berthomieu and M. Diaz. (1991) Modeling and verification of time dependent systems using time petri nets. *IEEE Transactions on Software Engineering*.
- [3] A. Campbell, G. Coulson, and D. Hutchison. (1994) A quality of service architecture. *ACM Computer Communications Review*.
- [4] D.R. Charinton and C.L. Williamson. (1989) Vmtp as the transport layer for high-performance distributed systems. *IEEE Communications Magazine*, pages 37-44.

- [5] I. Chrisment. (1994) Impact of alf on communication subsystems design and performance. In *Proceedings of the 1st Int. Workshop on High Performance Protocol Architectures*, INRIA, Sophia Antipolis.
- [6] D.D. Clark, M.L. Lambert, and L. Zhang. (1987) Netblt : a bulk data transfer protocol. rfc 998.
- [7] D.D. Clark and D.L. Tennenhouse. (1990) Architectural considerations for a new generation of protocols. In *Proceedings of ACM SIGCOMM*.
- [8] A. Danthine, editor. (1994) *The OSI95 Transport Service with Multimedia Support*. Springer Verlag. Research Reports ESPRIT, Project 5341, OSI95, Vol.1.
- [9] L. Delgrossi and J. Sandvoss. (1993) The berkcom-ii multimedia transport system.
- [10] M. Diaz, A. Lozes, C. Chassot, and P.D. Amer. (1994) Partial order connections. a new concept for high speed and multimedia services and protocols. *Annals of Telecommunications, Tome 49, n° 5-6*.
- [11] M. Diaz and G. Pays. (1994) The cesame project : Formal design of high speed multimedia cooperative systems. *Annals of Telecommunications, Tome 49, n° 5-6*.
- [12] M. Diaz and P. Sénac.(1994) Time stream petri nets for multimedia information. In *LNCS R. Valette (Ed), Int. Conf. on Application and Theory of Petri Nets*, Zaragoza.
- [13] C. Diot. (1995) Adaptive applications and qos guaranties. In *Invited paper to IEEE Multimedia Networking*, Aizu, Japan.
- [14] C. Diot, C. Huitema, and T. Turletti. (1995) Network conscious applications. In *HPCS workshop, Mystic*.
- [15] TENET group. (1994) *Recent and Current Research*. University of California, Berkeley, USA.
- [16] L. Henckel. (1993) Multimedia communication platform : Specification of the enhanced broadcast transport service. In *CIO RACE Project 2060*.
- [17] Protocol Engines Incorporated. (1992).
- [18] ISO/TC97/SC21/WG1-FDT/SG-B. (1987) Estelle, a formal description technique based on an extended state transition model. *Draft International Standard ISO/TC97/SC21-DP9074*.
- [19] T. Little and A. Ghafoor. (1990) Synchronization and storage models for multimedia objects. *IEEE Journal on Selected Areas in Comm.*, 8(3).
- [20] Inc. MIL 3. (1994) *OPNET documentation*. MIL 3 Incorporated / 34000 International Drive / Washington, DC 20008. OPNET documentation.
- [21] P. Sénac, M. Diaz, and P. de Saqui-Sannes. (1994) Toward a formal specification of multimedia synchronization scenarios. *Annals of Telecommunications, Tome 49, n° 5-6*.
- [22] B. Walter. (1983) Timed petri-nets for modeling and analysing protocols with time. In *Proceedings of PSTV, III, IFIP*.
- [23] N. Williams and G.S. Blair. (1994) Distributed multimedia applications : a review. *Computer Communications, vol.17, n° 2*.

PART SIX

Architectural Issues

A Performance Model for Integrated Layer Processing*

Bengt Ahlgren

Swedish Institute of Computer Science

Box 1263, S-164 28 Kista, Sweden

E-mail: Bengt.Ahlgren@sics.se

Abstract

Integrated Layer Processing is an implementation technique for data manipulation functions in communication protocols. The purpose of this technique is to increase communication performance. It reduces the number of memory accesses and thus relieves the memory bandwidth bottleneck. Integrated Layer Processing can however, in some situations, substantially *increase* the number of memory accesses, and therefore instead reduce performance. The main reason is contention for processor registers.

We present a performance model that captures the memory behavior of data manipulation functions for both integrated and sequential implementations. By comparing the model to measurements of real and synthetic data manipulation functions, we show that the model accurately predicts the performance. The model can be used to assess whether an integrated implementation will perform better or worse than a sequential implementation. The situations where integration would reduce performance can then be avoided without spending a lot of effort on a more complex integrated implementation.

Keywords

ILP, Integrated Layer Processing, performance modeling, protocol implementation techniques

*This work is supported in part by the CEC DG III Esprit LTR project 21926 HIPPARCH.

1 INTRODUCTION

The communication performance of workstation class computers is limited by the performance of their primary memory system (Druschel, Abbott, Pagels and Peterson, 1993; Smith and Traw, 1993). The reason is that software implementation of protocol data manipulation, such as checksum calculation, needs to access every byte of message data in memory. Since the access time of memory chips are decreasing at a much slower pace than CPU speeds are increasing (Patterson and Hennessy, 1994; Wulf and McKee, 1995), the bottleneck is getting more and more severe relative to the CPU performance of future computer systems.

Integrated Layer Processing, ILP, is an implementation technique for data manipulation functions in communication protocols presented by Clark and Tennenhouse (1990). The purpose with ILP is to relieve the memory bottleneck by reducing the number of memory accesses for message data and thus increase communication performance. The potential reduction comes from combining the manipulation functions of several protocol layers into a pipeline in a single processing loop. For each iteration of the loop the manipulations are run in succession on a small quantity of data, typically a word or a couple of words. Message data is pipelined via registers between the manipulations and need not be accessed from memory by each function.

ILP has been shown to considerably improve throughput of simple functions (Abbott and Peterson, 1993; Gunningberg, Partridge, Sirotkin and Victor, 1991; Partridge and Pink, 1993) and to improve throughput less for more complicated functions and when used in complete systems (Braun and Diot, 1995; Gunningberg, Partridge, Sirotkin and Victor, 1991).

In a previous paper (Ahlgren, Björkman and Gunningberg, 1996) we have shown that ILP also can substantially *decrease* performance compared to a conventional sequential implementation. The conclusion in that paper was that the use of CPU registers play a very important role for the performance gain, or loss, for an integrated implementation compared to a sequential. We found that when the aggregated state size of the data manipulation functions does not fit in processor registers, the integrated implementation quickly becomes much slower than the sequential implementation.

In this paper we continue to study the performance behavior of integrated and sequential implementations of data manipulation functions. Since the purpose of ILP is to reduce the number of memory accesses, we are particularly interested in the memory behavior and the role of the CPU registers. We present a performance model that predicts the performance of the two implementation techniques. The model calculates the number of clock cycles per message byte given a set of architecture parameters and a set of function parameters describing the computer system and the data manipulation functions, respectively. We show that the predicted performance from the model compare well with measurements.

The important contribution of our performance model is that it does not only consider memory accesses for message data, but also memory accesses for function state that does not fit in processor registers.

The motivation behind this work is to fully understand how different parameters affect the performance of the two implementation techniques. This knowledge can be useful for tools ("ILP compilers" (Braun and Diot, 1996b)) that automatically generate integrated implementations.

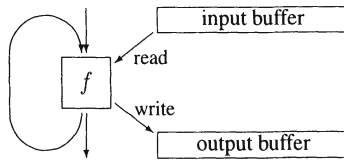


Figure 1 A data manipulation function “ f ”.

2 RELATED WORK

We have studied Integrated Layer Processing using a technique with synthetic data manipulation functions. These results are presented in a previous paper (Ahlgren, Björkman and Gunningberg, 1996). We concluded that the key factor is whether the aggregated function state fits in the available CPU registers or not. We showed experimentally that an integrated implementation can be slower than the corresponding sequential.

Abbott and Peterson (1993) also recognizes that “the key limiting factor is register pressure” when the number of integrated functions is increased. They also predict that the integrated implementation can be slower than the sequential but do not show it analytically or verify it experimentally. They present a performance model, but it only includes memory references to message data. In their model, the integrated implementation is therefore always faster than the sequential implementation. We compare our model with their model in a later section of this paper.

Integrated Layer Processing has also been studied by a number of other researchers (Braun and Diot, 1995; Braun and Diot, 1996a; Gunningberg, Partridge, Sirotkin and Victor, 1991). They have however not presented any performance model for comparing an integrated implementation with a sequential.

Previous work have paid quite a lot of attention to the practical problem of applying ILP to a real protocol stack (Abbott and Peterson, 1993; Braun and Diot, 1996a). We are aware of these problems but do not address them in our work. These problems are independent of whether integration as such has the potential of producing a faster implementation for a specific set of data manipulation functions or not.

3 DATA MANIPULATION FUNCTIONS AND ILP

Communication protocol processing can be divided in two parts: protocol control and data manipulation. The protocol control part processes the information in message headers and maintains the state needed by the protocol, for example, connection state. The control processing is usually specified and implemented as a finite state machine. The data manipulation part processes the data part of the messages. Examples of data manipulation functions are checksums, encryption and presentation encoding. They all have in common that they process each byte of the message data and possibly produce new data. This is illustrated in Figure 1. The data to be manipulated is stored in buffers. The input and output buffers are often the same buffer in a real implementation, but to be more general we illustrate them as separate buffers. The data manipulation function f is implemented as a loop which reads a block of data from the input

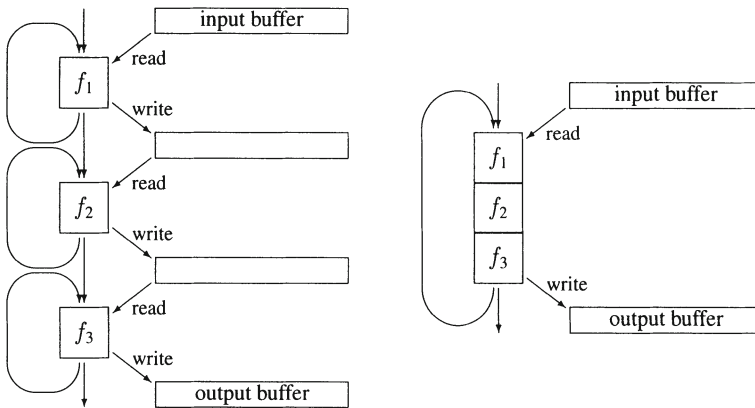


Figure 2 Composition of data manipulation functions, sequential (left) and integrated (right).

buffer, computes the function value and writes the result to the output buffer. The minimum size of the block depends on the function.

3.1 Sequential vs. integrated implementations

In a traditional implementation of a protocol stack, the data manipulation functions are executed independently in their respective layer, as illustrated to the left in Figure 2. We call this implementation a *sequential* implementation. Each function follows sequentially and reads and, possibly, writes all data in the buffer. The functions need not be immediately adjacent as in the figure. They can even be located in different address spaces. An example of this is the implementation of the TCP/IP protocol stack with an RPC protocol in a typical Unix system. It has presentation layer encoding in the application and the UDP or TCP checksum in the operating system kernel.

The idea with Integrated Layer Processing (Clark and Tennenhouse, 1990) is to combine the data manipulation functions of several protocol layers into one processing loop as illustrated to the right in Figure 2. The purpose is to avoid reading and writing the message data in the buffers several times. The performance gain has shown to be substantial (Abbott and Peterson, 1993). We call this implementation an *integrated* implementation. It reads a block of data from the input buffer and applies all functions in succession on the data. The assumption is that the transformed data is stored in processor registers between the functions and need not be accessed from memory by each function. After the last manipulation function the resulting block of data is written to the output buffer and the loop continues with the next block until the input buffer is finished. In this particular example with three functions each producing new data, the number of memory accesses for message data is reduced to $\frac{1}{3}$ of the sequential implementation.

3.2 Characterizing data manipulation functions

Before we can develop the model explaining and predicting the behavior of an integrated implementation compared to a sequential implementation, we first need a way to describe the

data manipulation functions at a suitable level of abstraction. Since the purpose with an integrated implementation is to save memory accesses, we must capture the characteristics that define the memory behavior of the function.

The following paragraphs describe the parameters we have identified as relevant.

Input data block The size of the input block is fundamental to a data manipulation function. The whole block must be present in order to do the processing. The function processes one block of data in each iteration. For example, the TCP/IP checksum has an input block size of 2 bytes, the DES encryption has 8 bytes and the MD-5 message digest has 16 bytes.

Output data block Depending on the particular function, the size of the output data block may be the same, less than or greater than the size of the input data block. A data compression function is an example of a function that produces less data than input. Some functions, like checksums, do not produce output data at all.

State Many data manipulation functions keep a state between iterations. This is the case for the TCP/IP checksum and the DES encryption algorithm in CBC mode. The checksum algorithm adds a 16 bit data block to the accumulated sum of previous blocks, i.e., the sum is the state. The state need to be stored somewhere—preferable in registers, since it is accessed for each iteration. The size of the state is therefore important for the memory behavior of the function. The final state is usually the result of functions that do not produce output data.

Constants Data manipulation functions can use constant values in its computation. A constant is explicit in the code and is accessed at least once per loop iteration. A constant is therefore a candidate for register allocation by the compiler. As such, constants compete for register space with the function state. A difference is that constants in memory need only be read, while the state is updated and must be both read and written. A constant can however be used more than once, so the compiler must analyze the number of references to be able to make a good register allocation.

Table look-ups Tables are similar to constants in that they contain constant values. The difference is that only a fraction of the entries in a table are accessed for each iteration. A table is also usually too large for the processor registers and must therefore be accessed from memory. The interesting parameters are the number of table lookups per iteration and the size of the table.

Temporaries A data manipulation function will use a number of temporary variables for intermediate results. The temporaries need not be explicit in the original source code. The compiler usually puts temporary variables in registers.

Instructions We distinguish between three logical categories of machine level instructions: *computation*, *memory* and *loop overhead*. The computation instructions are arithmetic-logic and control instructions used to compute the state or the output data of the data manipulation function. The memory instructions transfer data between the CPU registers and the memory or the cache. The loop overhead instructions consist of instructions for incrementing the buffer pointers, decrementing and testing the loop variable and a conditional branch back to the beginning of the loop.

The particular machine instructions and the number of them in an implementation depend both on the machine architecture and the compiler used to produce the code. The computation instructions, however, are pretty well defined by the data manipulation function. The number of memory instructions depend on register allocation, except for the accesses to the input and output data buffers which always are referenced in memory. The loop overhead instructions depend mainly on the processor instruction set, but also on the compiler.

Table 1 Model parameters.

Architecture parameters	
R	number of available CPU registers (word size)
A_C	access time of first level cache (clocks/word)
A_M	access time of memory (clocks/word)
CPI	cycles per instruction
L	loop overhead (instructions/loop)
C	CPU clock frequency
W	word size (bytes)
Function parameters	
n	number of functions
s_i	state size for function i (words)
a_i	computation instructions for function i per input block
b_i^{in}	input block size for function i (bytes)
b_i^{out}	output block size for function i (0 if no output)

4 THE PERFORMANCE MODEL

In this section we present a performance model for data manipulation functions. The model computes the cost in time per data byte both for an integrated implementation and for a sequential implementation of the same combination of data manipulation functions. The costs of the implementations can then be compared to see which performs best.

4.1 Model parameters

Table 1 lists the input parameters of the model. The parameters are of two kinds. The *architecture* parameters specify the characteristics of the target computer system. The *function* parameters specify the characteristics of the desired data manipulation functions.

The number of CPU registers (R) denotes the number of registers that the compiler has available for the function state. This parameter is thus compiler dependent in addition to being CPU dependent. We assume that this number is invariant when the other parameters of the model are varied. The memory access time (A_M) is the average access time for reading and writing memory sequentially through the cache(s) of the system. The effects of cache line fills and cache line write-backs are included in this average. We assume that the read and the write access times are the same.

The CPI parameter models the superscalarity of the system. A superscalar processor can execute more than one instruction in a single clock cycle. Non-superscalar processors have $CPI \geq 1$. Superscalar processors can have $CPI < 1$, but the value is usually very dependent on the mix of instructions.

The loop overhead (L) depends on the instruction set of the CPU, but can also depend on the compiler. As defined in the previous section, the loop overhead consist of incrementing two buffer pointers (or one pointer for checksum-like functions), decrementing and testing the loop variable, and a conditional branch back to the beginning of the loop.

To make the model as simple as possible, yet still capturing the desired behavior, we have

chosen to not include constants and tables of the function parameters described in the previous section. We believe that this does not significantly affect the modeled memory behavior of the respective implementation techniques. The constants have almost the same influence on the memory behavior and total performance as the function state has, since the constants also compete for the available registers and overflows on the stack. The tables add a random component of scattered memory accesses which is very similar, if not identical, for both implementation techniques.

Another limitation in the model is the selection of the output block size. A function must either not produce output at all (checksum-like functions), or produce the same amount of output as input, i.e., the output block size must be the same as the input block size.

4.2 Model formulas

The purpose of the performance model is to accurately model the register, cache and memory behavior of the integrated and sequential implementations of a set of data manipulation functions. The number of memory accesses are therefore central in the model. There are two kinds: accesses for message data, denoted DA (*data accesses*), and accesses for function state, denoted SA (*state accesses*). Both of these are expressed as the number of word accesses (load or store) per byte of message data. DA for the integrated case is simply:

$$DA^{ILP} = \frac{2}{W} \quad (1)$$

that is, one load and one store for each word of data. We here assume that there is at least one function producing new data. We also assume that functions with a block size that is less than the word size still can read and write data a word at a time.

In the sequential case we have to add the loads and stores for each function:

$$DA^{SEQ} = \frac{1}{W} \times \sum_{i=1}^n \begin{pmatrix} b_i^{out} = 0 & : & 1 \\ b_i^{out} > 0 & : & 2 \end{pmatrix} \quad (2)$$

If b_i^{out} is zero, the function only reads the message data (1 access). Otherwise it writes as well (2 accesses).

The number of state accesses for the integrated implementation is:

$$SA^{ILP} = \frac{2}{b^{ILP}} \times \begin{cases} S^{ILP} \leq R & : & 0 \\ S^{ILP} > R & : & S^{ILP} - R \end{cases} \quad (3)$$

The important factor in this equation is the parameter R , the number of CPU registers available for function state, and its relationship to S^{ILP} , which is the aggregated state of all functions:

$$S^{ILP} = \sum_{i=1}^n s_i \quad (4)$$

If the whole state fits in the registers, there are no memory accesses. Otherwise, part of the state has to be stored in memory. This is the key behavior of the model that is expressed by Equation 3.

We assume that the state in memory has to be loaded from and stored back to memory once per iteration of the ILP loop, thus the “2” in the numerator of the fraction in Equation 3. b^{ILP} in the denominator is the block size common for all functions in the integrated implementation. It is the least common multiple of the block sizes of all manipulation functions:

$$b^{\text{ILP}} = \text{lcm}(b_i^{\text{in}}, i = 1 \dots n) \quad (5)$$

The SA formula for the sequential implementation is a sum over the functions:

$$SA^{\text{SEQ}} = \sum_{i=1}^n \left(\frac{2}{b_i^{\text{in}}} \times \begin{cases} s_i \leq R & : 0 \\ s_i > R & : s_i - R \end{cases} \right) \quad (6)$$

The expression inside the sum is essentially the same as for the integrated case, but the state and block sizes are for the respective original functions instead of for the integrated function.

Equations 3 and 6 simplify the case where there is a function whose state alone overflows the registers. The model assumes that the code can be arranged so that only one load and one store are required for each word of overflowed state. This is not true for all functions. It depends on the specific details of the function algorithm. For functions where more memory accesses than one load and one store are necessary, the model underestimates the number of accesses. The underestimation may be larger, but never less, for the integrated implementation compared to the sequential, because the overflowed state in the former is never smaller than the overflowed state of any individual function in the latter.

The message data and state accesses just defined can either be satisfied by the processor cache or the real memory. We assume that the state accesses will always be present in the first level cache of the processor. This is a reasonable assumption, since the state is accessed on every iteration of the loop. The message data is on the other hand not always present in the cache. We are therefore interested in both the case when data are present in the (first level) cache and the case when data has to be fetched from memory. The behavior of a real system will be somewhere in between. Below we present one formula each for these two cases. The formulas are used for both the sequential and integrated cases with the respective substitution of the above defined formulas for DA and SA .

The number of clock cycles per byte of message data in the cached case is:

$$\text{clocks (cached)} = CPI \times \left(\sum_{i=1}^n \frac{a_i}{b_i^{\text{in}}} + LO \right) + A_C \times (DA + SA) \quad (7)$$

The second term is the time to access the message data and the state from the cache (A_C is the cache access time parameter). The first term is the number of clock cycles for the non-memory instructions. The sum is the number of computation instructions per byte for all functions. LO is the number of instructions of loop overhead. It is computed with the following formulas:

$$LO^{\text{ILP}} = \frac{L}{b^{\text{ILP}}} \quad (8)$$

$$LO^{\text{SEQ}} = \sum_{i=1}^n \frac{L}{b_i^{\text{in}}} \quad (9)$$

The sum of the computation instructions and the loop overhead is then multiplied by CPI , the cycles per instruction parameter, to get the time to execute all instructions. The total formula then represents the number of clock cycles to execute the instructions and access the memory.

The non-cached, or memory, case is similar to the cached case, but the data accesses are multiplied by A_M , the memory access time:

$$\text{clocks (memory)} = CPI \times \left(\sum_{i=1}^n \frac{a_i}{b_i^{\text{in}}} + LO \right) + A_M \times DA + A_C \times SA \quad (10)$$

Finally, if the time in seconds is desired instead of clock cycles, the number of clock cycles are divided by the CPU clock frequency:

$$\text{time per byte} = \frac{\text{clocks}}{C} \quad (11)$$

5 COMPARISON WITH MEASURED RESULTS

In this section we compare the modeled performance to the measurements from a previous paper (Ahlgren, Björkman and Gunningberg, 1996). In that paper we measured the performance of integrated and sequential implementations of “synthetic” data manipulation functions. The synthetic functions are generated from the same set of function parameters as we use in the model. The synthetic implementations process data and use memory and registers according to the parameters, but does not produce any “useful” output. Synthetic functions makes it easy to study the behavior when one parameter (e.g., the size of the state) is varied and all other parameters are kept constant.

5.1 The Abbott & Peterson loop

The first comparison is both with a real implementation of the BSWAP/PES/CKSUM function combination used by Abbott and Peterson (1993) (see their Figure 2) and with a synthetic implementation with the same characteristics. The BSWAP/PES/CKSUM loop consists of a byte-swap, a “pseudo-encryption” and the Internet checksum. We have made one change to their original integrated implementation. We use a block size of two words which saves the “if” statement and gives better performance. We discuss the block size further in Section 6.

Table 2 presents the measured performance of the real and synthetic implementations compared with the modeled performance for three computer systems. The modeled performance values compare well to the measured real implementation for all three systems. The largest deviation is about 10 % (for the HP, sequential with warm cache). Tables 3 and 4 lists the model parameters used in the comparison.

The available CPU registers have been determined by looking at the machine code produced by the respective compilers from synthetic functions with varying state size. The compilers used are the stock “cc” compilers for both operating systems (SunOS 4.1.x and HP-UX A.9.5). The SunOS compiler is not particularly good at register allocation. The value of the register parameter therefore varies with the situation.

Of the memory access figures, the value for the SPARCstation 2 is calculated from the actual memory access time. For the SPARCstation 20/71, the value is calculated from the maximum

Table 2 Comparison of the measured performance of the real and synthetic implementations with the modeled performance using the BSWAP/PES/CKSUM loop.

System	impl. type	Performance, ns/Byte			
		cold cache		warm cache	
		Seq.	ILP	Seq.	ILP
HP 9000-735/99	real	95	49	54	37
	synthetic	95	49	54	37
	modeled	87	52	49	37
SPARCstation 2	real	344	174	230	125
	synthetic	351	186	236	137
	modeled	372	178	247	128
SPARCstation model 20/71	real	78	39	54	32
	synthetic	82	39	57	33
	modeled	81	40	58	31

Table 3 Parameter values for the experimentation systems.

Architecture parameter		HP 735	SS-2	SS-20
R	available CPU registers	13	8	8
A_C	cache access (clocks/word)	1	3	1
A_M	memory access (ave. clocks/word)	4	7	2.4
CPI	cycles per instruction	1	1	0.5
L	loop overhead (instr/loop)	1	5	5
C	clock frequency (MHz)	99	40	75
W	word size (bytes)	4	4	4

sustained memory bus speed, and is thus the lower bound on the memory access time. The value for the HP 9000-735/99 is a good guess that needs to be further verified.

The CPU of both the HP and the SPARCstation 2 have one integer unit, so the CPI parameter can not be lower than one. The CPU in the SPARCstation 20/71 has two integer units and therefore has the possibility to execute the computation part of our functions at twice the clock speed. The author was surprised to find that the model produced the best output with CPI set to exactly 0.5. This indicates that the processor actually use both integer units fully in this situation.

Table 4 Parameter values for the BSWAP/PES/CKSUM loop.

Function parameter		BSWAP	PES	CKSUM
s_i	state size (words)	0	0	1
c_i	constants (words)	3	2	2
a_i	instructions per input block	5	6	4
b_i^{in}	input block size (bytes)	4	8	4
b_i^{out}	output block size	4	8	0

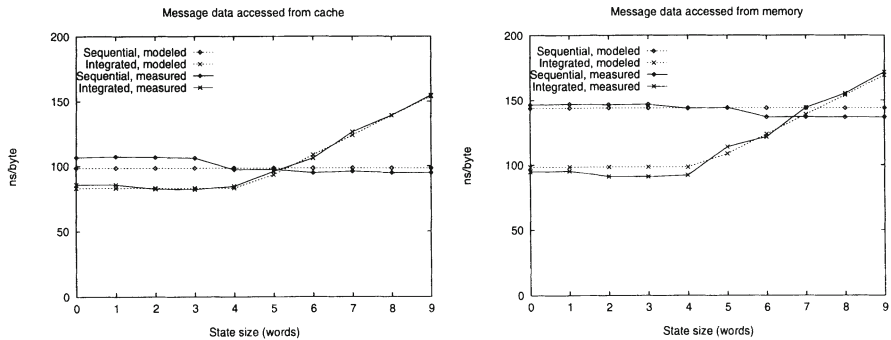


Figure 3 Comparison of measured performance with modeled performance for the HP 9000-735/99.

As previously mentioned, for simplicity we chose not to include the constants parameter in the model. It does not affect the result in this case, because all of the state and the constants fits in the available registers. But it leads to a weakness in the comparison. Since the state fits in the registers, the register exhaustion part of the model is not exercised. In the next section we therefore also compare with synthetic loops with varying state sizes.

5.2 Synthetic loops

The second comparison is with measurements of synthetic data manipulation functions on the HP 9000-735/99. We have varied the state size and the number of functions. The other function parameters have been kept constant: 3 functions, 4 byte input and output block sizes and 10 computation instructions per input block. The architecture parameters used are those for the HP in Table 3.

As can be seen in Figure 3, the measured and modeled performance correspond very well. We can see that the model captures the register exhaustion just as intended for the integrated implementation when the state size grows above 4 words. For a state size of 5 words, the aggregated state is $3 \times 5 = 15$ words, which does not fit in the available 13 registers.

6 MODEL ANALYSIS

How do different factors affect the performance of an integrated implementation compared to a sequential? In this section we will discuss the role of the parameters in the model and how they affect the performance of the two implementation techniques.

6.1 Abbott and Peterson's model

But first we compare the performance model with the model presented by Abbott and Peterson (1993) (see Figure 4). Their model computes the number of clock cycles for loop overhead, computation and data accesses. This basic structure is very similar to our model. We also count

	Loop overhead	Computation	Data access
seq:	$3 \times num_layers + computation + data_access_cycles \times (num_layers + num_write_layers)$		
ilp:	3	$+ computation + data_access_cycles \times 2$	

Figure 4 The Abbott and Peterson model, estimated cycles to manipulate one data word.

the cycles for loop overhead and computation instructions. If we set the *CPI* parameter to 1, we basically get the same result. Both models also compute the cost for accessing the message data.

The main difference is that their model does not include memory accesses for function state. It can therefore not be used to study the situation we are interested in when the aggregated function state does not fit in the processor registers. As a consequence, in their model the integrated implementation always performs better than the corresponding sequential implementation.

Another difference is that we have chosen to have separate expressions for the cached and non-cached cases, whereas they capture cache access time, cache hit rate and memory access time in one parameter (*data_access_cycles*).

6.2 Function parameters

State size and number of functions The function state sizes and the number of functions defines the aggregated state size for the integrated implementation. When the aggregated state size exceeds the number of available registers, the excess state will typically be allocated on the stack. This results in additional memory referencing instructions.

If the aggregated state fits in registers, the integrated implementation will always be faster than the sequential. In the model the number of state accesses will then be zero, and both the number of data accesses and the loop overhead will be higher for the sequential implementation.

This is the general performance behavior that the model expresses and which is illustrated by the examples in Figure 5. The respective surface shows the performance in clock cycles per byte of message data for different number of functions and state sizes. The data shown is dominated by the memory and cache performance. There are only three instructions of computation per function and one instruction of loop overhead. The memory access time used to produce the graph in Figure 5b is 5 CPU clock cycles per word.

In the left part of both graphs, where the state size is small, the modeled performance of the integrated implementation is better than the sequential implementation. There are knees on all curves at the point where the aggregated state size completely fills the available registers (10 in these examples). To the right of the knees, there are additional memory references for the part of the function state that overflows the registers. These references are all assumed to be satisfied by the cache. The intersections between the integrated and the sequential surfaces are the points where these additional state accesses for the integrated implementation cost the same as the additional message accesses and loop overhead in the sequential implementation.

Block size The block size of the data manipulation functions may not seem interesting at first from a performance point of view. Previous research have focused on the practical aspect on how to integrate functions with different block sizes.

Abbott and Peterson (1993) advocate *word filters* with a uniform one-word block size. Func-

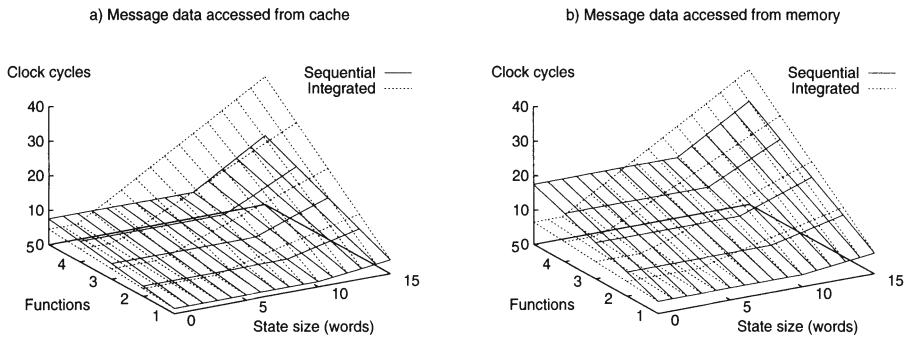


Figure 5 Modeled performance for different state sizes (per function) and number of integrated functions.

tions having a larger natural block size are in their model recoded to collect the required number of one-word blocks before doing its computation. The following functions are then repeatedly invoked with a one-word input.

Braun and Diot (1996a) advocate that the least common multiple of the functions is used as the block size of the integrated implementation. We agree with their approach, but with a different motivation. The main drawback with the word filter approach is that it introduces additional conditional branches and state variables in the inner integrated loop. In a simple experiment with the previously mentioned BSWAP/PES/CKSUM combination of functions, we received between 25 % (for cached message data) and 35 % (for non-cached message data) improvement in performance when we increased the block size to two words from the original one word.

There are also other considerations for the block size in the case when the function state is too large for the processor registers. When the block size is increased, the additional loads and stores of the function state is amortized over the larger block size. This effect is seen in Figure 6. The slope above the knees is decreasing rapidly when the block size increases. For the case in the figure, the integrated implementation is faster than the sequential when the block size reaches 32 bytes, regardless of state size.

In this example, the input and output block sizes are the same. The number of available registers are 10 for a block size of 4 bytes. For all larger block sizes, the number of available registers is decreased with the number of additional registers needed for the larger input block. The other model parameters used to generate this graph are: 3 instructions per 4 bytes of message data, 3 functions, 1 cycle cache access time, 1 CPI and 1 instruction of loop overhead. The figure shows the case when message data is in the cache. When the message data is not present in the cache, the absolute difference between the surfaces becomes larger with a constant, but the shapes of the surfaces are the same.

Computation instructions In a previous paper (Ahlgren, Björkman and Gunningberg, 1996) we showed that the number of computation instructions does not affect the absolute performance difference between an integrated and a sequential implementation. The model also has this behavior, since the computation instructions add the same number of clock cycles for both implementation types.

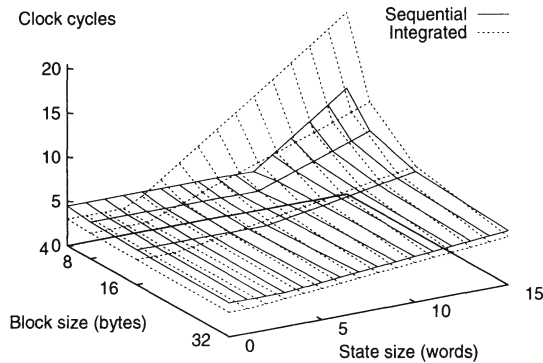


Figure 6 Modeled performance for different state sizes (per function) and block sizes.

6.3 Architecture parameters

Number of registers The number of processor registers available for function state defines the threshold point when the excess function state is allocated on the stack. Processors with more available registers increase the applicability of the ILP technique to more functions and to functions with larger state.

As an example, the MD-5 message digest function has a state (the chaining variables) that needs 4 registers. MD-5 has, however, a 64 byte block size, which potentially uses another 16 registers, and 64 unique constants.

Another example is the Safer family of encryption algorithms. It has an 8 byte block size, but processes the message data a byte at a time and therefore needs 8 registers to hold the input block. The whole expanded encryption key is used for each input block. The expanded key is 104 bytes for the minimum recommended 6 “rounds” of the algorithm, which potentially use 26 registers. Actually, because of the byte operations, one register per byte is desirable.

Both of these functions are examples where the CPU registers can not hold everything that is needed in the inner processing loop. If these functions are integrated with another function having state, the registers will overflow even more.

Memory access time and cache hit rate The memory access time and the cache hit rate are very important for the performance difference between the integrated and sequential implementations. The slower the memory and the lower the cache hit rate, the more is gained with integration.

The actual performance experienced in a real system depends heavily on the cache hit rate for accesses to message data. Sequential implementations are more sensitive to varying cache hit rates than integrated implementations, since sequential implementations access data several times.

Conversely, in a real system integrated implementations result in lower cache hit rates than sequential implementations. This is simply the result of that the message data is accessed only once.

In extreme cases with low cache hit rates (for message data) and slow memory, the integrated implementation may always be faster than the sequential implementation, also in the cases when the function state overflows the registers. This is illustrated in Figure 7.

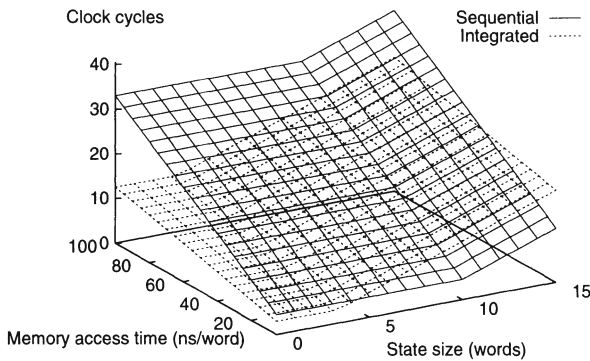


Figure 7 Modeled performance for different state sizes (per function) and memory access time.

Cache access time The cache access time determines the slope of the performance curve for increasing state sizes above the register threshold point. A reasonable assumption is that the cache access time scales with the CPU clock frequency of future processors. This means that the cached performance scales proportionally with higher clock frequency for both the integrated and sequential implementations.

CPU clock frequency When the clock frequency of the processor increases, the memory speed get relatively slower. As we have previously mentioned, one of the main motivations behind ILP is the fact that processor speeds are increasing at a much higher rate than memory latency is decreasing. This will make ILP more and more applicable.

A higher clock speed will have the same effect on the relative performance of an integrated and a sequential implementation as a higher memory access time has.

7 CONCLUSIONS

We have presented a performance model for integrated and sequential implementations of data manipulation functions in communication protocols. We have shown that the model correctly captures the behavior of memory accesses both to message data and to function state that overflows the CPU registers. The model explains the complex relationship between the data manipulation functions, the computer architecture and the resulting performance of the integrated and sequential implementations. It accurately predicts the performance of the integrated as well as the sequential implementation of the BSWAP/PES/CKSUM function combination and a number of synthetic data manipulation functions.

The model can predict whether an integrated implementation will perform better or worse than a sequential implementation of the same set of data manipulation functions. It can therefore be used to determine whether or not effort should be spent on a more complex integrated implementation.

Future work includes comparing the model with measurements of more complex data manipulation functions such as the MD-5 message digest and the Safer encryption.

ACKNOWLEDGMENTS

The author wishes to thank Per Gunningberg and Mats Björkman for the discussions during the development of the model and for comments on drafts of this paper.

The author also thanks the anonymous reviewers for valuable comments.

REFERENCES

- Abbott, M.B. and Peterson, L.L. (1993) Increasing network throughput by integrating protocol layers. *IEEE/ACM Transactions on Networking*, 1(5):600–610, October.
- Ahlgren, B., Björkman, M. and Gunningberg, P. (1996) Integrated layer processing can be hazardous to your performance. In *IFIP Protocols for High Speed Networks*, Sophia-Antipolis, France, October 28–30.
- Braun, T. and Diot, C. (1995) Protocol implementation using integrated layer processing. In *SIGCOMM '95 Conference Proceedings*, pages 151–161, Cambridge, MA, USA, August 28–September 1. *ACM SIGCOMM Computer Communication Review*, 25(4).
- Braun, T. and Diot, C. (1996a) Performance evaluation and cache analysis of an ILP protocol implementation. *IEEE/ACM Transactions on Networking*, 4(3):318–330, June.
- Braun, T. and Diot, C. (1996b) Automated code generation for integrated layer processing. In *IFIP Protocols for High Speed Networks*, Sophia-Antipolis, France, October 28–30.
- Clark, D.D. and Tennenhouse, D.L. (1990) Architectural considerations for a new generation of protocols. In *SIGCOMM '90 Conference Proceedings*, pages 200–208, Philadelphia, Pennsylvania, September 24–27. *ACM SIGCOMM Computer Communication Review*, 20(4).
- Druschel, P., Abbott, M.B., Pagels, M.A. and Peterson, L.L. (1993) Network subsystem design. *IEEE Network*, 7(4):8–17, July.
- Gunningberg, P., Partridge, C., Sirotkin, T. and Victor, B. (1991) Delayed evaluation of gigabit protocols. In *Proceedings of the 2nd MultiG Workshop*, Electrum, Stockholm-Kista, Sweden, June 17.
- Partridge, C. and Pink, S. (1993) A faster UDP. *IEEE/ACM Trans. on Networking*, 1(4), August.
- Patterson, D.A. and Hennessy, J.L. (1994) *Computer Organization and Design: The Hardware/Software Interface*. Morgan Kaufmann Publishers. ISBN 1-55860-281-X.
- Smith, J.M. and Traw, C.B.S. (1993) Giving applications access to Gb/s networking. *IEEE Network*, 7(4):44–52, July.
- Wulf, W.A. and McKee, S.A. (1995) Hitting the memory wall: Implications of the obvious. *Computer Architecture News*, April.

BIOGRAPHY

Bengt Ahlgren received his MSc degree in Computer Science from Uppsala University in 1989. He joined the PhD program at the Department of Computer Systems at Uppsala University in 1990 and is currently finishing his PhD thesis with expected graduation in March 1997.

He has been employed full time as a researcher at the Swedish Institute of Computer Science (SICS) since 1989. The first few years he worked on communication security. Later work has been focused on high speed communication and the performance of protocol implementations and host/network interfaces.

An Architecture for Active Networking

Samrat Bhattacharjee, Kenneth L. Calvert, Ellen W. Zegura
Networking and Telecommunications Group
College of Computing, Georgia Institute of Technology,
Atlanta, GA 30332.
email: {bobby,calvert,ewz}@cc.gatech.edu

Abstract

Active networking offers a change in the usual network paradigm: from passive carrier of bits to a more general computation engine. The implementation of such a change is likely to enable radical new applications that cannot be foreseen today. Large-scale deployment, however, involves significant challenges in interoperability, security, and scalability. In this paper we define an active networking architecture in which users control the invocation of pre-defined, network-based functions through control information in packet headers.

After defining our active networking architecture, we consider a problem (namely, network congestion) that may benefit in the near-term from active networking, and thus may help justify migration to this new paradigm. Given an architecture allowing applications to exercise some control over network processing, the bandwidth allocated to each application's packets can be reduced in a manner that is tailored to the application, rather than being applied generically. Our results show that the ability to gracefully adapt to congestion makes a good case for active networking.

Keywords

active networks, congestion control, MPEG, early packet discard

1 INTRODUCTION

As the cost of computing power decreases, it is worthwhile to consider the benefits of adding computing to various types of systems, either to enhance services or to trade off against other costs such as time, bandwidth and storage. *Active Networking (AN)* refers to the addition of *user-controllable* computing capabilities to data networks*. With active networking, the network is no longer viewed as a passive mover of bits, but rather as a more general computation engine: information injected into the network may be modified, stored, or redirected as it is being transported. Obviously, such a capability opens up many exciting possibilities. However, active networking also raises a number of issues, including security, interoperability and migration strategy. All of these are influenced in large part by the active networking *architecture*,

*In this paper we focus on packet- and cell-switched networks. We follow Tennenhouse and Wetherall [7] in adopting the term active networking; the differences between our approach and theirs are elucidated later.

which defines the interface between the users and the capabilities provided by the network.

In this paper, we consider an approach to active networking that generalizes traditional packet switching. In our approach, users can select from an available set of functions to be computed on their data, and can supply parameters as input to those computations. The available functions are chosen and implemented by the network service provider, and support specific services; thus users are able to influence the computation of a selected function, but cannot define arbitrary functions to be computed. This approach has some benefits with respect to incremental deployment as well as security and efficiency: AN functions can be individually implemented and thoroughly tested by the service provider before deployment, and new functions can be added as they are developed.

2 AN ARCHITECTURE FOR ACTIVE NETWORKING

2.1 A Generic Model of Packet Processing

The network consists of switching *nodes*, which are connected via *links*. In this simple model, nodes don't do anything except process the packets received on their incoming links; processing an incoming packet may result in one or more packets being transmitted on outgoing links.

More precisely, the *state* of a node comprises the following pieces:

- An *input queue* of packets. Packets received on any link are placed in the input queue.
- For each outgoing link, an *output queue* containing packets to be transmitted on that link.
- A collection of *generic state information*. This represents long-lived information maintained at the node, such as routing tables or virtual-circuit switching tables.

We postulate that each node in the network supports a particular set of functions, each of which has a unique identifier. Each packet contains a set of *headers*, which specify (i) the identifier of one or more functions to be applied to the packet; and (ii) parameters to be supplied to those functions. When the packet is processed, the function identified by each header is applied, resulting in updating of the node's state and possibly modification of the rest of the packet. Thus we postulate that for each function identifier f , and each parameter value p for function f , there is a particular subset of the node's generic state information that is *relevant* to f and parameter p . Functions cannot modify or use parts of the node state that are not relevant.

Each node repeatedly performs the following:

```

Remove a packet  $M$  from the input queue;
while (more functions need to be applied to  $M$ ):
    Let  $f, p$  be the function ID and parameter from the next header of  $M$ ;
    Let  $g$  be the state component relevant to  $f$  and  $p$ ;
    Invoke function  $f$  on  $M$ , with  $p$  as parameter:
        (optionally) Modify  $M$ ;
        (optionally) Update  $g$ ;
        (optionally) Queue messages for output;

```

Traditional networking functions can be characterized as node-processing functions in this model. For example, when an IP datagram containing destination address X is received by a router node (none of whose addresses is X) over an Ethernet, the router node examines two of the datagram headers, which identify the “Ethernet function” and the “IP forwarding function”. The Ethernet function checks the error-control code and removes the Ethernet header, while the IP forwarding function determines which outgoing link the packet should be forwarded on, and adds the appropriate link-level header for that link.

2.2 From Packet Forwarding to Active Networking

We define active networking to be extension of the set of functions that can be invoked at a network node beyond those required to simply move bits from place to place. The basic idea of our approach to active networking is the incremental addition of user-controllable functions, where each function is precisely defined and supports a specific service.

In general, the introduction of new AN functions of the type we have in mind involves specification of the following:

- The *identifier* associated with the function.
- The *parameters* associated with the function, and the method of encoding them in a packet.
- The *semantics* of the function. Ideally, function semantics would be given in a standard notation such as LOTOS [3], SDL [4], or another notation developed specifically for the purpose. A *standard environment*, comprising support services such as private state storage and retrieval, access to shared state information (e.g. routing tables), message forwarding primitives, etc., would provide a foundation on which new AN functions services could be built.

In this view of AN, addition of a new function to a network node would be the responsibility of the network service provider. As with current networks, once a function is specified, each provider or vendor would be free to imple-

ment the functionality in a manner consistent with the specification. This approach corresponds roughly to the way new features are deployed in the public switched telephone network today: users have the option of provisioning various features implemented and deployed by the service provider.

There is another approach to extending the set of available functions at a node: adding a single very powerful (Turing-complete) function, which interprets its parameter as a program and then executes that program with the packet and the relevant state as inputs. This single function thus extends the set of functions computable at a node to include all computable functions. The “active capsule” approach of Wetherall and Tennenhouse [7] is based upon this idea. In this approach, the interface between the user and the active network is a programming language, with well-defined syntax and semantics.

2.3 An AN Implementation

We have implemented several AN functions in an IP routing architecture. We defined an “Active Processing” IP option `IPOPT_AP`, which specifies a particular AN function in IP routers. The `IPOPT_AP` option is a tuple of the form $\langle f, m, \alpha \rangle$. The identifier f identifies the active processing function and m is used to retrieve cached state associated with the current flow. α is a variable (possibly null) sequence of arguments to the function f . We augmented an existing SunOS IP kernel to support an in-kernel active processing router. We implement an associative memory function M , which can be used to retrieve state (e.g. queues) associated with the current flow. We define an AN function dispatch table Δ , such that $\Delta[i]$ is the address of the i th active processing function. We modified the IP option processor to execute $\Delta[f](M(m), \alpha)$ for each datagram that specifies an AN IP option. We simulate congestion by defining a “virtual router memory limit.” The datagram discard routine is initiated as this virtual router memory is exhausted. To let the router queues grow, we service the router queues at predefined output rates — this is used to simulate various output link rates.

3 PROGRAMMABLE CONGESTION CONTROL

3.1 Operating Model

A *flow*, for our purposes, is a sequence of packets all having the same source and destination, from the point of view of a node somewhere in the network. Thus a flow might consist of packets traveling between a single pair of endpoints, or it might be the aggregation of a set of lower-level flows. We assume that a flow is identified by a label of some kind in the network protocol header.

Generically, programmable congestion control operates as follows: *Based*

on triggers that indicate congestion control should take place, flow state is examined for advice about how to reduce quantity of data.

The important components of this generic model are the *triggers* responsible for initiating congestion control, the *flow state* that contains the specific advice for this flow, and the *reduction techniques* defined by the network and made available to the users. An important feature of this model is its consistency with traditional best-effort service. That is, a flow provides *advice* about what to do with its data. The network node is not required to take the advice, and may apply generic bandwidth reduction techniques.

In its most powerful and general form, congestion control might reduce bandwidth requirements by applying *transformations* to reduce the quantity of data at a congestion point. Such transformations will be application-specific and computationally complex (not to mention the significant issues they raise regarding interaction with end-to-end mechanisms such as error detection and sequence numbering). We therefore focus on the special case of *intelligent discard* of data. Using a natural extension of packet-level discard [1], we allow applications to define units based on application semantics, with aim of discarding the entire unit if any portion must be discarded. It should be noted that for this scheme to be most effective, buffering is required to collect application data units to ensure that the “tail” will not be dropped. Further, the end-to-end reliability mechanisms (if any) affect the optimal definition of a unit. The most benefit derives from dropping units that are equivalent to the reliable retransmission units; dropping a unit that is smaller or larger may result in unnecessary retransmission.

Given that bandwidth reduction will occur by discarding units, a question arises as to which unit (within a flow) to discard. In the most simple case, there is no choice: when the congestion indication trigger occurs, a fixed unit (typically the one currently being processed) is subject to discard. More efficient network behavior is possible, however, if we expand the set of data from which we choose to discard. We consider congestion control advice that indicates how to make such a choice. One can think of this advice as indicating priority or some other policy by which to discriminate across data in the same flow. Making use of this advice clearly requires that the network node have access to a collection of data within a single flow. Thus, these mechanisms will involve storing and manipulating flow data before it leaves the node, e.g., while sitting in a per-flow queue from which packets are periodically selected for output by a scheduling mechanism.

3.2 Example Application and Mechanisms

To illustrate specific mechanisms and evaluate performance, we focus on the use of congestion control advice to improve the best-effort service provided to MPEG. For our purposes, the important feature of an MPEG stream is

that it consists of a sequence of frames of three types: I, P and B. Coding dependencies exist between the frames, causing P and B-frames to possibly require other frames in order to be properly decoded. Each I-frame plus the following P and B frames forms a group of pictures (GOP), which can be decoded independently of the other frames.

The specific components of the programmable congestion control are implemented as follows:

- **Source-attached advice.** We consider mechanisms in which the source identifies “units” such that the unit will be discarded if any portion of the unit must be dropped.

The **Partial Packet Discard** mechanism is for baseline comparisons; it defines each IP fragment to be a unit. Fragments are discarded if they cannot be buffered in the output queue.

Our **Frame Level Discard (EMD)** mechanism defines a unit to be an MPEG frame. The advice given is to queue a datagram if and only if its corresponding frame can be queued in its entirety. We maintain state for each frame that is being discarded or buffered, and use this state information to decide, in constant time, to buffer or discard a particular datagram.

We further consider a mechanism that identifies dependencies between units. Our **Group of Picture (GOP) Level Discard** maintains state about the type of frame discarded. In case an I frame has been discarded, we discard the corresponding P and B frames as well.

- **Choice among units.** We consider one policy for making choices amongst units. When an I frame is too large to be accommodated in the output queue, and the queue contains P and B frames such that their combined sizes are greater than that of the I frame, then such P, B frames are discarded, and the I frame transmitted.
- **Triggers.** Finally, we consider two types of triggers. In the first, we detect and respond to congestion only when data arrives that cannot fit in the output queue. All three mechanisms mentioned above use this trigger. We also consider an “early” trigger, following the Early Packet Discard of Romanow and Floyd [6]. This trigger detects and responds to congestion when the output queue occupancy exceeds a certain threshold. Our **Early MPEG Discard** mechanism combines the GOP Level Discard with an early congestion trigger.

4 EXPERIMENTAL METHODOLOGY

4.1 Topology and Data streams

The experimental topology is shown in Figure 1. The bandwidth of the link between the source and the router, and between the congestion producing host

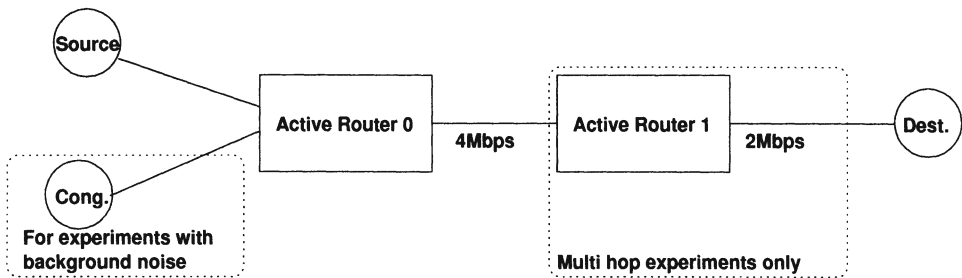


Figure 1 Experimental topology

MPEG stream	Avg. Frame (bytes)	Avg. I Frame (bytes)	Avg. GOP (bytes)	Avg. Y SNR
B0	8177	20222	81782	21.00
F0	9256	36755	92568	23.05
M0	8970	36272	89713	23.86
T0	4174	18623	41750	25.60

Table 1 MPEG stream statistics

and the router is varied in the experiments. The source host runs Solaris 2.5.1. All other hosts run SunOS 4.1.3; the routers executing our modified SunOS 4.1.3 kernel. The physical connectivity is provided by 155 Mbps ATM links. The output queues of the routers 0, 1 have 48K, 64K of memory, respectively.

Four different MPEG streams were subjected to the same tests. Each stream consisted of 120 frames in 12 GOPs. Particulars about the streams are provided in Table 1. For experiments with background traffic, the routers were congested by replaying packet traces with average cumulative bandwidth of 2.15 Mbps.

4.2 Metrics

We evaluate the results of our experiments using the following criteria.

- **SNR of received stream.** Under congestion, IP packets carrying the MPEG data may be discarded at intermediate routers. We evaluate the received stream by computing the signal-to-noise ratio (SNR) of the received stream – in cases when frames are dropped, we use the last correctly decoded frame as a substitute for the dropped frame. The SNR is computed

with respect to the original YUV files which were used to create the transmitted stream.

- **Fraction of I Frames received.** The fraction of I frames that are received provide a good measure of the quality of the resultant MPEG stream.
- **Fraction of bytes that were eventually discarded at the receiver.** Due to the inter-dependent frame structure of MPEG, frames can only be decoded if all other frames that they depend on have also been received. We measure the fraction of data that is received but cannot be used due to incomplete frames, or missing related frames.

4.3 Destination Feedback Mechanism

Our active mechanisms can be used in conjunction with source adaptation. We implemented a simple form of flow control over UDP. The flow control mechanism has three parameters: the feedback resolution F , the reaction rate R and the source increment S . The mechanism operates on the principles of linear increase and exponential decrease as follows. The receiver sends feedback whenever it determines that at least F frames have been transmitted since it last sent feedback. If all F frames were received correctly, the receiver sends an ACK, otherwise it sends a NACK. For each NACK, the sender cuts its rate in half. Upon receiving R consecutive ACKs, the sender increases its rate by S .

5 RESULTS

5.1 Fixed Rate Sources

In Figure 2, we consider the simplest case of a single congested node, a fixed rate source, and no feedback from the destination. On the x-axis, we vary the fixed source rate from 2 Mbps to 24 Mbps, corresponding to increasing overload on the 4 Mbps link between the router and the destination. On the y-axis we plot the average value of the signal-to-noise ratio for the Y component of the MPEG data. Each curve corresponds to a different trigger/advice mechanism, as described in Section 3.2.

This plot is primarily included so that the reader has a baseline to keep in mind when we consider other combinations of feedback, background traffic and multi-node experiments.

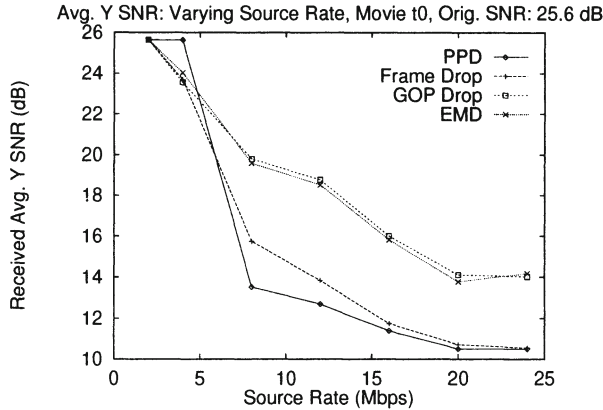


Figure 2 Avg. Y SNR at destination, with fixed source rate

5.2 Sources with Destination Feedback

We consider cases in which the destination provides feedback to the source as described in section 4.3. The end nodes implement the flow control algorithm with the following parameters. The source rate is initialized to 1 Mbps, and constrained to less than 8 Mbps. The source increment S is 2 Mbps, and the reaction rate R is $\lfloor 40/F \rfloor$, where F is the feedback resolution.

In Figure 3, the GOP level discard and EMD mechanisms consistently maintain an average SNR above 21 dB, while transmitting nearly 80% of the I-frames (Figure 4). No data transmitted via GOP level discard, or EMD is discarded at the receiver (Figure 5). Due to oscillations in the source rate, and indiscriminate discard at the intermediate router, the average SNR maintained by the PPD algorithm is 5–10 dB less in all cases (compared to GOP discard, and EMD). Both the PPD and the Frame level discard algorithm transmit data that is discarded at the receiver, thereby wasting network resources. The Frame level discard algorithm only transmits complete frames, thus, the data discarded at the receiver is due to the transmission of undecodable frames. In case of PPD, data is also discarded due to receipt of partial frames*.

5.3 Addition of Background Traffic

The addition of background traffic leads to further congestion of the output link at the router. Under this scenario, with choice enabled, the GOP level

*The MTU for our experiments was 4096 bytes. The losses due to individual frame fragmentation would be greater if the MTU had been 1500 or 576 bytes.

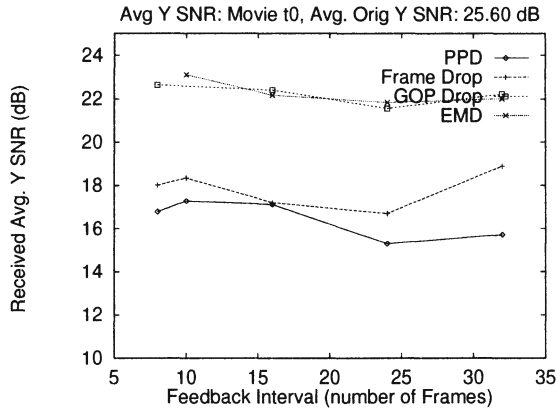


Figure 3 Avg. Y SNR at destination providing feedback

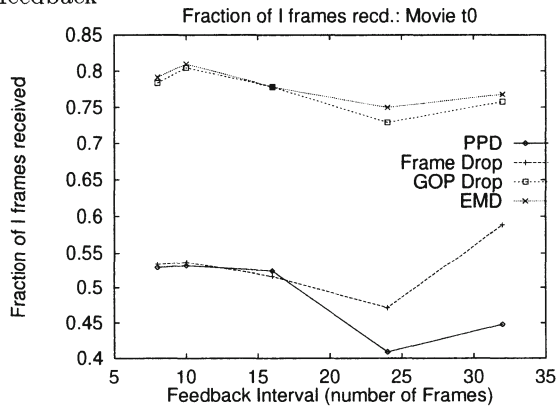


Figure 4 Fraction of I frames received at destination

discard and EMD mechanisms are able to maintain an average SNR of 19–22 dB (Figure 6). Under the same circumstances, the average SNR maintained by PPD varies between 10–17 dB. The GOP discard and EMD mechanisms are able to transmit 70–80% of the I frames, while 5–50% of the I frames are transmitted using PPD (not shown). Between 30 and 95% of the data transmitted by PPD is discarded at the receiver. Interestingly, a larger fraction of data transmitted by the Frame level discard algorithm is discarded in this case (with background traffic) than without (not shown). This is due to the smaller amount of buffering available to the MPEG stream (because of the background traffic). Thus, the choice policy has fewer frames to work with, and in many cases P and B frames are transmitted, as discarding them would

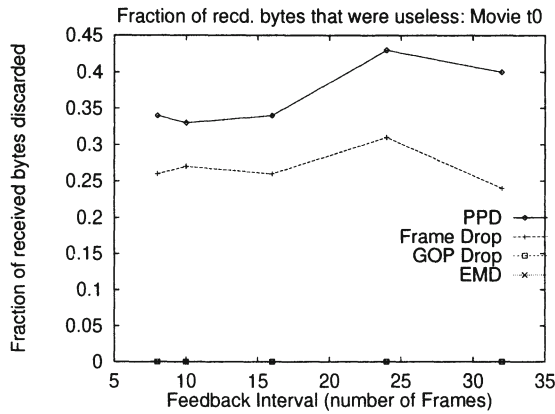


Figure 5 Fraction of received bytes discarded at destination

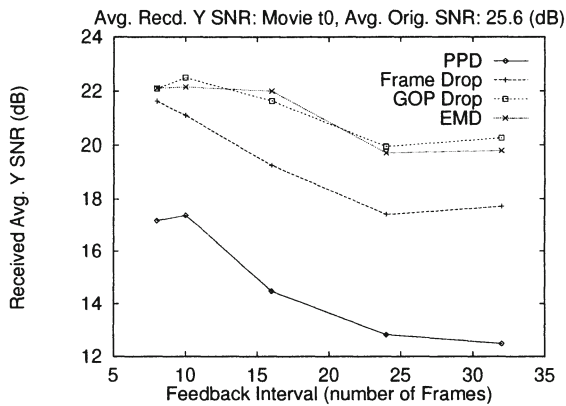


Figure 6 SNR at dest. (with background traffic, and feedback)

not create enough space to transmit another I frame. Note that in these cases, the GOP level discard and the EMD mechanisms do not transmit the P and B frames, and thus conserve network resources.

5.4 Multi-Node Experiments

We extend the topology to include one additional router. Figure 7 shows the fraction of I frames received after the MPEG stream encounters two congested routers. The output link of the first router is restricted to 4 Mbps, and the

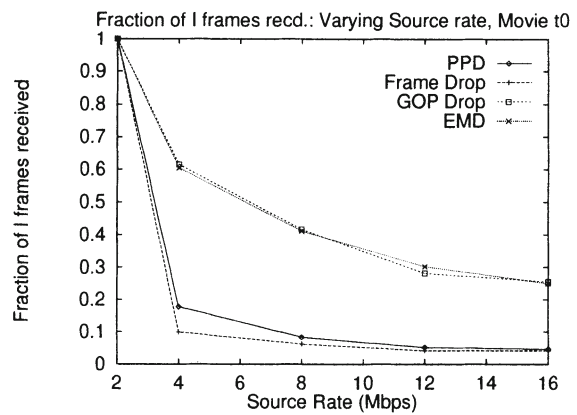


Figure 7 Fraction of I frames received (multi-AN hops, no feedback)

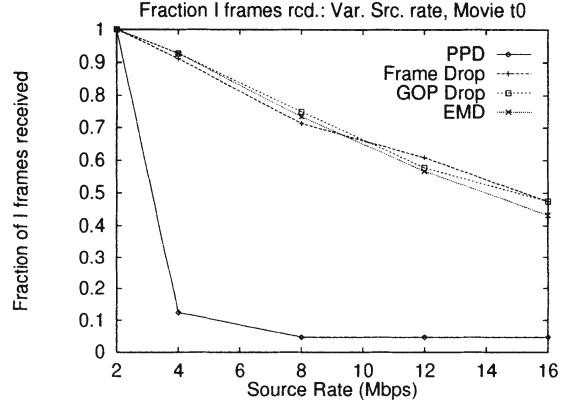


Figure 8 Fraction of I frames received (multi-AN hops, Choice policy enabled)

output link of the second router is restricted to 2 Mbps. As expected, 30—50% more I frames were received using GOP level discard, and EMD. A similar increase was noted for the received signal level. The drop in fraction of I frames transmitted (and SNR) is more gradual for the GOP level discard, and EMD mechanisms. It should be noted that between 70—80% of the data transmitted by the Frame level discard, and PPD mechanisms were eventually discarded by the receiver.

When we add the choice policy to the same scenario (Figure 8), the mechanisms other than PPD transmit a higher fraction of the I frames. A similar increase is noted in the received signal level as well. The greatest impact is on the Frame level discard algorithm — which in some cases, can transmit

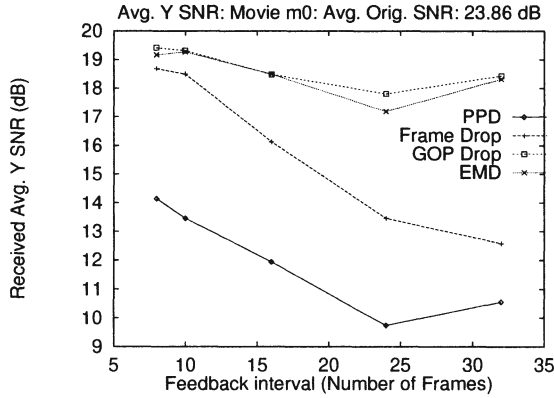


Figure 9 Avg. Y SNR at destination (multi-AN hops, dest. feedback)

over 80% more useful data using the queue manipulation techniques. Also, less than 10% of data transmitted by the Frame level discard algorithm is eventually discarded at the receiver.

In Figure 9 we consider a multi-node experiment, with destination feedback, with the choice policy enabled. Within favorable feedback intervals (feedback per 4-12 frames), the mechanisms other than PPD sustain an average SNR of 18-19 dB. Using the same feedback mechanism, the average SNR maintained by the PPD mechanism varies between 7-14 dB. Once again, more than 80% of the data transmitted by PPD is discarded at the receiver.

5.5 Detailed Evaluation of Transmitted Streams

(a) Time varying SNR

In Figure 10, we consider a specific example, and evaluate the received data on a frame-by-frame basis. The destination provides feedback (as described in section 4.3) every 10 frames, and the stream passes through two routers. The plots above the x-axis represent the SNR of the received frames. The points below the x-axis represent indices of complete frames received. Under GOP discard, 163 of 220 total frames are received, while under PPD, 148 total frames were received. However, the GOP discard mechanism was able to transmit 22 of the possible 23 I Frames, while PPD was able to transmit only 12 I Frames. From Figure 10, it is clear that PPD is susceptible to periods of catastrophic losses (as seen in indices [60-120], and [160-230]). During the same period, GOP discard experiences heavy frame losses as well; but is able to transmit the crucial I frames (except for frame index 100), thus preserving some picture quality. This leads to a visibly graceful degradation of the signal (indices [90-110]) - and recovery from signal loss is much quicker than PPD (e.g indices [190-210]).

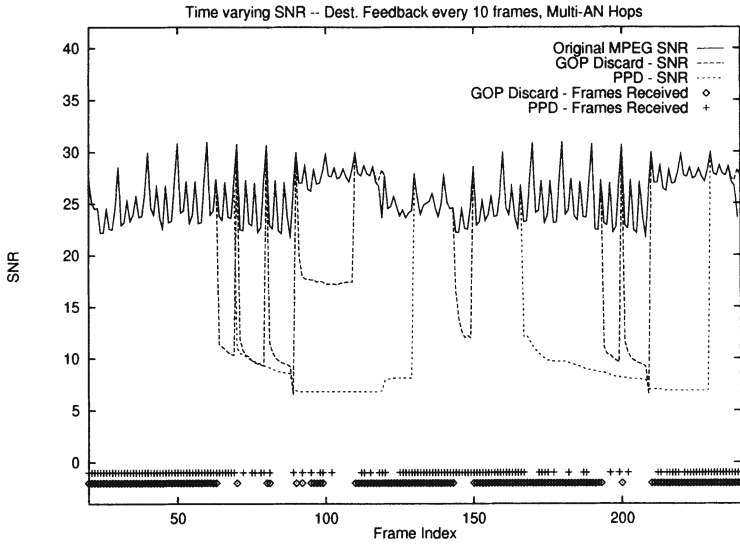


Figure 10 Per Frame SNR of received stream

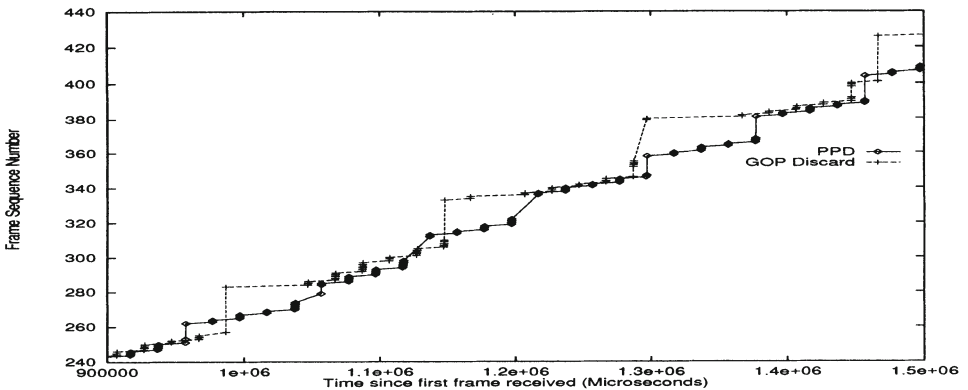


Figure 11 Fixed Rate Source: Timing of received datagrams

(b) Transmission Latency

In Figure 11, we consider the delay added by the processing. The source rate is set to 8 Mbps. Considerable detail about the system behavior over time is represented by Figure 11 which plots the time elapsed (in microseconds) since the first datagram is received for datagrams with sequence numbers from 240 to 440. Note that one can easily discern the points at which GOPs are discarded, corresponding to a (nearly) vertical line segment indicating a range of unreceived datagram sequence numbers. PPD contains periods of discard that occur more frequently, but tend to last for a shorter period of time. An important point to note is that the mechanisms do not introduce any additional buffering delay (compared to PPD). As IP packets are pro-

cessed, an $O(1)$ check is made whether this packet can be transmitted, or not. Frames, or GOPs, are not reassembled in the routers; this would have introduced buffering delays. Delays, if any, are introduced during the computation phase of the packet acceptance algorithm. Under the choice policy, this computation time may include time taken to search the queue. The search time is $O(\text{Output Queue Size}/\text{Packet Size})$. However, the only packets that encounter this delay are packets that would otherwise been discarded. This is because the choice is only activated when a packet — specifically an I frame — cannot be accommodated in the output queue.

6 CONCLUSIONS

The premise of active networking is that users can benefit from enhanced network functionality. We have presented a simple approach to AN in packet-switched networks, in which users can control the application of a set of supported network-based functions to their data. Our approach permits new functions to be developed and deployed over time, and is backward compatible in that not all users need be aware of the active functionality in the network, and not all nodes need support the same set of functions.

7 ACKNOWLEDGEMENTS

The trace data used was collected by Steve Klivansky, Scott Register, Hans Werner-Braun, and Amarnath Mukherjee. David Haynes implemented the signal-to-noise ratio computation.

REFERENCES

- [1] G. Armitage and K. Adans. Packet reassembly during cell loss. *IEEE Network Magazine*, September 1993.
- [2] F. Bonomi and K. Fendick. The rate-based flow control framework for the available bit rate ATM service. *IEEE Network Magazine*, March/April 1995.
- [3] E. Brinksma. An introduction to lotos. In *Proc. 7th IFIP WG 6.1 Workshop on Protocol Specification, Testing and Verification*, 1987.
- [4] CCITT. Recommendations z.101 to z.110. Blue Book, 1988.
- [5] V. Jacobson. Congestion avoidance and control. In *ACM SIGCOMM '88*, 1988.
- [6] A. Romanow and S. Floyd. Dynamics of TCP traffic over ATM networks. *IEEE Journal on Selected Areas in Communications*, May 1995.
- [7] D. L. Tennenhouse and D. J. Wetherall. Towards an active network architecture. In *Multimedia Computing and Networking '96*, January 1996.

The Strategy of Traffic Dispersion

E. Gustafsson

Royal Institute of Technology, Dept. of Teleinformatics

KTH-Electrum/204, S-164 40 KISTA, SWEDEN,

Phone +46 8 7521498, Fax +46 8 7511793, Email evag@it.kth.se

G. Karlsson

Swedish Institute of Computer Science

Box 1263 S-164 28 KISTA, SWEDEN

Phone +46 8 7521577, Fax +46 8 7517230, Email gk@sics.se

Abstract

Traffic dispersion means spreading the traffic from a source over multiple independent paths, transmitting it in parallel through the network. The strategy reduces the effects of bursts in the traffic, and may hence improve the network performance in terms of reduced queuing delay. So far, there have been several reports on traffic dispersion, but to our best knowledge, there has been no in-depth investigation of how the strategy affects different types of traffic under various conditions. This paper focuses on the basic properties of traffic dispersion by defining the strategy and investigating it from the source's as well as from the network's point of view. We investigate the influence of dispersion on Poisson traffic, traffic generated by two-state Markov chains and traffic generated by the chaotic FPD-map. The results indicate a large potential of traffic dispersion to provide fast, secure and fault-tolerant transmission for highly bursty traffic sources.

Keywords

Traffic dispersion, ATM, traffic characteristics, queuing performance, network performance.

1 INTRODUCTION

The broadband integrated services digital network, B-ISDN, is intended to carry various types of traffic, such as voice, data and video. Satisfying the various user performance requirements will put stringent demands on the network to be flexible, reliable and still cost-efficient, that is, to obtain high resource utilization. Some of the difficulties imposed on the network are discerned when considering the traffic behaviour of potential B-ISDN users. While ordinary telephone traffic traditionally has been modelled by Poisson processes, recent studies indicate the characteristics that stem from data and multimedia applications to be vastly different. Essentially, studies have shown this type of traffic to exhibit strong correlation over long time periods, Leland et al. (1994), Paxson and Floyd (1995).

The asynchronous transfer mode, ATM, is the recommended network architecture for B-ISDN. Succinctly described, it combines the circuit-switched routing of telephone systems with the statistical multiplexing of packet switching, by establishing a virtual connection through the network before transmission. The information is then sent over the connection in fixed-size packets called cells. In order to better utilize the network resources, ATM employs statistical multiplexing, which means that the capacity of a transmission link in the network is statistically shared among the connections traversing it. This implies that the demand for capacity

occasionally may exceed the available resources. Surplus traffic could temporarily be buffered at the link access, but would result in buffer overflow and loss of data if continued inordinately.

The probability of information loss due to buffer overflow is highly dependent on the correlation in the multiplexed traffic streams Li (1989). Long-term correlation and extended traffic bursts complicate resource allocation because of the difficulties to guarantee performance. Given a connection, the correlation in a traffic stream may be lowered by spreading the traffic in time, so called shaping. The amount of shaping is however limited by the permissible delay that the information transfer may suffer, and the residual correlation in a shaped traffic stream may consequently still be strong. Multiplexing of such correlated streams at low probability of loss would require unreasonable low utilization of the network capacity, and excessively large buffers.

Another method to reduce the correlation in a traffic stream is to spread the traffic from a source over multiple disjoint paths, thereby sending it in parallel through the network, Figure 1 (a). We call this technique *traffic dispersion*, and we define the dispersion factor N to be the number of paths over which the traffic from a source is spread. Dispersion may be used on several network levels; over paths, links or multiple channels within a link. In this study, we refer to traffic dispersion as dispersion of packets over disjoint paths, that is, paths that do not share links statistically. Given a certain number of paths, there are different ways to spread the traffic. *Cyclic dispersion* is the spreading of packets from a source in round-robin order over the paths (Figure 1 (b)). If the correlation in the traffic stream is monotonously decreasing, this spreading strategy minimizes the correlation in the traffic stream on each path (see Section 2.2.2). *Sequential dispersion* means spreading sequences of L consecutive packets (Figure 1 (c)). A sequence length $L=1$ corresponds to cyclic dispersion. Sequential dispersion with $L>1$ does not necessarily reduce the correlation in the traffic stream, and if the sequence length is large, the performance may appear even worse with than without dispersion (see Section 3.1). In both cyclic and sequential dispersion, the order of the paths is known in advance, which facilitates the spreading and resequencing mechanisms. *Dynamic dispersion* spreads the traffic dynamically over the paths, according to the current network load. This makes it possible to avoid congested or heavily loaded paths, but requires continuously updated network information in return. Furthermore, this strategy does not consider the correlation in the traffic stream and may result in strongly correlated traffic on one or several of the paths, possibly congesting paths that were originally clear. Contrary to the two earlier methods, dynamic dispersion is reactive rather than preventive, and if all users were to spread their traffic dynamically, the result could be an unstable operation. The analyses in this study focus on the preventive dispersion strategies.

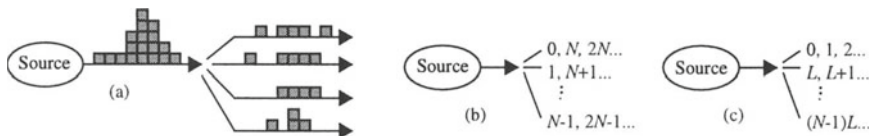


Figure 1 Illustration of traffic dispersion (a), and cyclic (b) and sequential (c) dispersion of the packets generated by a traffic source.

During the last years, traffic dispersion has been discussed in various terms and for different purposes, Gustafsson and Karlsson (1997). Throughout the studies, traffic dispersion has shown to give performance improvement in terms of better queuing behaviour and reduced loss. The use of independent paths also makes loss of

information on one path independent of losses on other paths, and consequently, forward error correction can be successfully used to increase network reliability. An additional advantage of traffic dispersion is that it enhances the network security by making eavesdropping on parallel connections simultaneously practically impossible. There exist however, to our best knowledge, no thorough investigation of how dispersion affects the behaviour of the traffic from different types of traffic sources, nor a comparison of the queuing performance under different load conditions. In order to motivate more advanced studies on traffic dispersion, its basic properties need to be defined, investigated and reported.

The aim of the work reported in this paper is to do such a study. We start in Section 2 by describing the system model used in the studies, including the different types of traffic sources used. In Section 3 we continue by investing the queuing behaviour for each traffic source. This Section also includes a performance comparison of cyclic and sequential dispersion, and an investigation of how large part of the traffic that must be dispersed in order to make the effects of dispersion show. Section 4 discusses the optimum dispersion factor for a call, and Section 5 concludes the paper.

2 SYSTEM MODEL DESCRIPTION

2.1 One-stage multiplexer model

In order to investigate the effects of traffic dispersion on some different traffic types, we use a one-stage multiplexer model according to Figure 2 (a) and (b). The case without dispersion is illustrated in Figure 2 (a), by a traffic source which generates traffic into a FIFO queue with fixed service rate. Next, the model is extended to spread the traffic from a source over N queues. This strategy will however reduce the amount of traffic entering the queue during a certain time interval. In order to keep the mean arrival rate to each queue constant, independent of N , we assume N independent identically distributed sources, each spreading its traffic over N queues (Figure 2 (b)). The amount of traffic arriving at each queue is thus kept constant, while the traffic behaviour changes due to the effects of dispersion. We define the utilization factor ρ in a queue to be the mean arrival rate multiplied by the queue service time. The buffer size is in the remainder of this paper considered infinite.

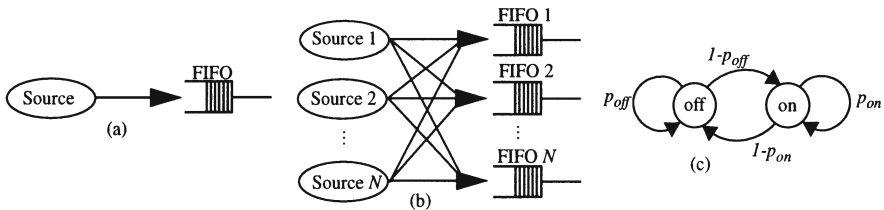


Figure 2 A one-stage multiplexer, consisting of traffic sources generating traffic into a number of FIFO queues. The case of no dispersion is shown in (a), while (b) illustrates dispersion. The state diagram of a two-state Markov chain is shown in (c).

2.2 Traffic source models

2.2.1 Poisson traffic

Poisson traffic is traffic which arrives according to a Poisson process, that is, the interarrival times between packets are independent and exponentially distributed. The Poisson process has been commonly used to model the arrival of telephone calls

during some time interval in a telephone network. A Poisson process with intensity λ is a process $X = \{X(t); t \geq 0\}$, where $X(t)$ denotes the number of arrivals of the process by time t , and

$$Pr\{X(t)=i\} = \frac{(\lambda t)^i}{i!} e^{-\lambda t}, \quad i = 0, 1, 2, \dots \quad (1)$$

The mean arrival rate is λ packets/time unit, and the variance of the arrival rate is λ . Since the arrivals of a Poisson process are mutually independent, the correlation in the traffic stream is zero.

2.2.2 Two-state Markov chain source

The most commonly used on-off source is based on a two-state Markov chain (Figure 2 (c)). The chain is alternating between an on state and an off state, and while in the on state, it generates traffic at peak rate h packets/time unit. The model may be seen as a crude model for data communications, and has been used with good accuracy to model speech after removal of silence periods.

In the following calculations, the peak rate of the source is set to $h = 1$ packet/time unit. Define $\varepsilon(i)$ as the number of packets generated within the i th time unit; $\varepsilon(i)$ is thus 0 or 1. The mean and variance of the chain are then given by

$$\bar{\varepsilon} = \frac{1 - p_{off}}{2 - p_{on} - p_{off}} \text{ and } \sigma^2 = \bar{\varepsilon}(1 - \bar{\varepsilon}). \quad (2)$$

$$\text{As in Li and Mark (1990), } \varepsilon(i) \text{ is normalized: } u(i) = \frac{\varepsilon(i) - \bar{\varepsilon}}{\sigma}. \quad (3)$$

The correlation between $u(i)$ and $u(i + n)$ is then given by

$$r(n) = E\{u(i + n)u(i)\} = (p_{on} + p_{off} - 1)^{|n|} = \phi^{|n|}. \quad (4)$$

The correlation is monotonously decreasing, and cyclic dispersion, which maximizes the distance between two consecutive packets on each path, hence minimizes the correlation. With cyclic dispersion, the correlation sequence becomes

$$r_d(n) = E\{u(iN + nN)u(iN)\} = r(nN) = \phi^{|nN|}. \quad (5)$$

The sojourn time the two-state Markov chain stays in a state is geometrically distributed with mean

$$\bar{T}_{on} = \frac{1}{1 - p_{on}} \text{ and } \bar{T}_{off} = \frac{1}{1 - p_{off}} \text{ respectively.} \quad (6)$$

Some of the numerical examples in this paper corresponds to transmitting voice traffic over ATM. The transition probabilities of the chain are then assigned values according to the mean length of talkspurts and silent periods, for example 1366 ms and 1802 ms, Brady (1968), and 227 ms and 596 ms, Lee and Un (1986) respectively. Transmitting voice over 64 kbit/s links makes an ATM cell (48 bytes of information) correspond to 6 ms, and the transition probabilities of a corresponding two-state

Markov chain are for the first example $p_{on} = 1-6/1366$ and $p_{off} = 1-6/1802$, and for the second $p_{on} = 1-6/227$ and $p_{off} = 1-6/596$.

2.2.3 The Fixed Point Double Intermittency map

Recent studies have shown tele and data communications traffic to exhibit long-range dependence, Leland et al. (1994), Paxson and Floyd (1995). Modelling traffic with such characteristics requires a new generation of traffic models, like for instance the fixed point double intermittency map, Pruthi (1995). The FPD-map is a chaotic map in the form of an on-off source, where the distribution of the time spent in each state can be varied from geometric to heavy-tailed. The nature of the distribution depends on the Hurst parameter of the source, and the FPD-map can be used to generate long-range dependent traffic.

3 QUEUING BEHAVIOUR

In this Section, we present the effects of dispersion in calculated and simulated queuing results. The question of cyclic or sequential dispersion is investigated in Section 3.1, and then the queuing behaviour is studied for the different source models. We start by discussing independent arrivals from a Poisson process in Section 3.2, continue with short-term correlated traffic generated by two-state Markov chains in Section 3.3 and finish in Section 3.4 by considering traffic generated by the chaotic FPD-map. Section 3.5 deals with the question of how many users must employ dispersion in order to make the positive effects of it show.

3.1 Cyclic or sequential dispersion?

In the following, we use the one-stage multiplexer model from Figure 2 (b) to investigate and compare the effects of cyclic and sequential dispersion. The simulation results in Figure 3 show how the mean queue size changes with increasing sequence length. The results presented in Figure 3 (a), for Poisson traffic sources, show that the mean queue size increases with the sequence length, and the larger dispersion factor, the faster increase. There is always a possibility that several sources will send traffic to the same queue simultaneously, thereby increasing the instantaneous peak arrival rate to the queue. As the sequence length increases, each source generates traffic continuously to each queue during a longer time interval. Several sources sending traffic into the same queue may hence cause traffic bursts of high peak rate. As a consequence of increased instantaneous peak arrival rate and increased correlation in the traffic, the mean queue size increases, even if the queue utilization factor is kept constant (see Section 3.3.2).

Figure 3 (b) shows the results from strongly correlated traffic generated by two-state Markov chains. In this case, the mean queue size increases practically linearly with the sequence length, and the larger the N , the faster the increase. For a large enough sequence length, dispersion aggravates the queuing behaviour, compared to the non-dispersed case. This may be explained by the same discussion as above.

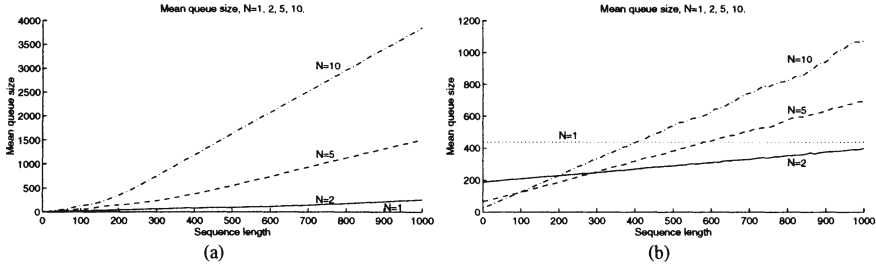


Figure 3 The mean queue size changing with the sequence length for Poisson traffic (a) and traffic generated by Markov chains with $p_{on}=1-6/1366$, $p_{off}=1-6/1802$ (b). The results are for $N=2, 5$ and 10 , and the mean queue size without dispersion is shown in (b), while in (a), it is too small to show. The utilization factor is 0.86 .

In essence, sequential dispersion may introduce bursts in the traffic, thereby aggravating the queuing behaviour. For a large enough sequence length, sequential dispersion performs worse than non-dispersed traffic, with respect to the mean queue size. Additional results, which are not presented here, show the standard deviation of the queue size to behave similarly. In the remainder of this paper, we therefore assume cyclic dispersion.

3.2 Queuing with Poisson traffic

3.2.1 Calculated mean queue size

To an $M/D/1$ queuing system, traffic arrives according to a Poisson process, and is served in the single server with a constant service rate. If we let λ be the mean arrival rate in packets/time unit, d the service time in time units and $\rho = \lambda d < 1$, the mean queue size, at moments of departure in equilibrium, is, Kleinrock (1975)

$$\bar{q} = \frac{\rho^2}{2(1-\rho)}. \quad (7)$$

Denote the time between arrival $i-1$ and i to the queue by X_i . The probability density function of X_i is then

$$f_{X_i}(x) = \lambda e^{-\lambda x}, \quad \forall i > 0 \quad (8)$$

and the mean interarrival time between packets is $1/\lambda$ time units. If the traffic from a source is cyclically dispersed with dispersion factor N , the arrivals to one of the queues corresponds to every N th arrival from the source. With $N=2$, the time between two adjacent arrivals to the queue is $Y = X_i + X_{i+1}$, the mean arrival rate is $\lambda/2$ packets/time unit, and the probability density function of Y is gamma distributed according to

$$f_Y(y) = \int_{-\infty}^{\infty} f_{X_i}(x) \cdot f_{X_{i+1}}(y-x) dx = \int_0^y \lambda e^{-\lambda x} \cdot \lambda e^{-\lambda(y-x)} dx = \lambda^2 y e^{-\lambda y}. \quad (9)$$

As the arrivals to the queue are still independent, the system becomes a $G/G/1$ queuing system. For this system, the mean waiting time in the queue, W , can be calculated using the spectral solution to Lindley's integral equation, described in Kleinrock (1975). In order to make a fair comparison of the queuing behaviour for different degrees of dispersion, ρ is kept constant. The distribution of the interarrival times to one queue is according to (9), and the distribution of service time is given by the Dirac delta function according to $\delta(x - 2d)$. Following the calculations in Kleinrock (1975), the mean waiting time and mean queue size are obtained as

$$W = \frac{\rho^2}{2\lambda(1 - \rho)} \text{ and } \bar{q} = \frac{\lambda}{2} \cdot W = \frac{\rho^2}{4(1 - \rho)}. \tag{10}$$

Repeating the calculations for different N gives the mean queue size for different degrees of dispersion. Figure 4 shows the calculated values and the corresponding simulated values.

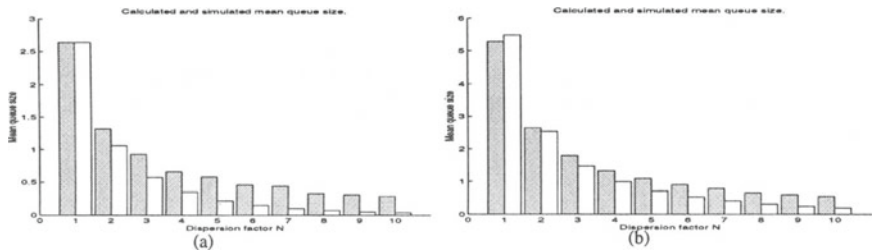


Figure 4 Calculated (shaded bars) and simulated mean queue size on one path, for different degrees of dispersion. The traffic is generated by a dispersed Poisson process, and the utilization factor is 0.86 (a) and 0.92 (b).

3.2.2 Simulated queuing behaviour

The results discussed above were for one source, spreading its traffic cyclically over N queues. In the following, the model with N sources spreading the traffic over N queues is employed (Figure 2 (b)). The simulated mean queue size and the standard deviation of the queue size in such a system are presented in Figure 5. These results show that the minimum mean queue size and standard deviation are obtained with a dispersion factor of approximately five. For a dispersion factor larger than five, even though the queue size of each dispersed source still decreases, as shown in Figure 4, superposing N dispersed sources causes the queue size to increase. Additional results, which are not presented here, show that for a lower utilization (smaller ρ), the optimum N is smaller than five, while a higher utilization makes the optimum N larger.

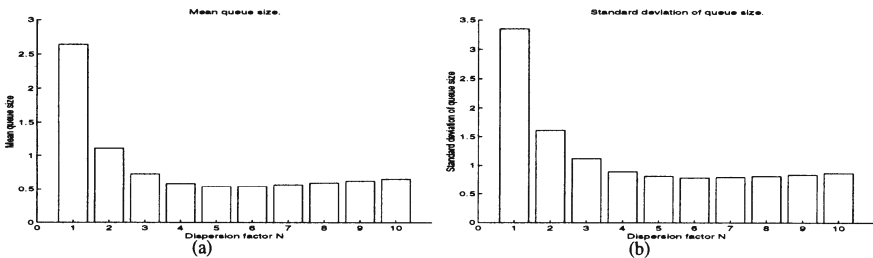


Figure 5 Mean queue size and standard deviation of the queue size for N Poisson sources generating traffic into N queues. The utilization factor is 0.86.

Figure 6 shows an example of how the queue size varies in time when using dispersion. The decrease in queue size agrees with the results from both Figure 4 and Figure 5. Thus, for Poisson arrivals, dispersion provides a means to reduce the mean queue size as well as the variance of the queue size. The results indicate the optimum dispersion factor to be in the interval [2, 5].

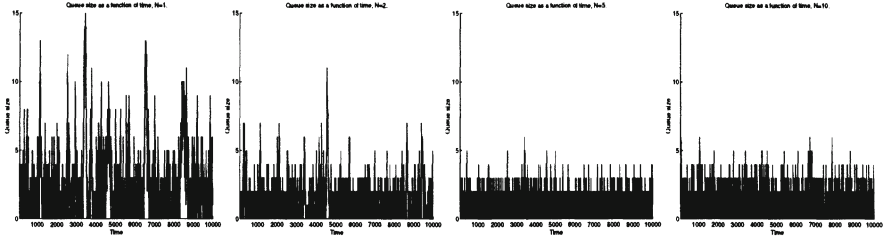


Figure 6 Example of queue size as a function of time for the on-stage multiplexer model with Poisson traffic, for $N=1, 2, 5$ and 10 . The utilization factor is 0.86 .

3.3 Queuing with traffic from two-state Markov chains

3.3.1 Calculated mean queue size

The power spectral density of a cyclically dispersed two-state Markov chain source is obtained as the discrete Fourier transform of the correlation sequence (5):

$$S(q) = \sum_{n=-\infty}^{\infty} r_d(n) e^{-j2\pi qn} = \frac{1 - \phi^{2N}}{1 + \phi^{2N} - 2\phi^N \cos(2\pi q)} \quad (11)$$

Since $|\phi| < 1$, $S(q) \rightarrow 1$ as $N \rightarrow \infty$. That is, when increasing the number of paths, the power spectral density will converge consistently with the definition of white noise. It is thereby clear that cyclic dispersion effectively reduces the correlation in the traffic generated by this type of source. Figure 7 shows the power spectral density of the packets on one path for different degrees of dispersion.

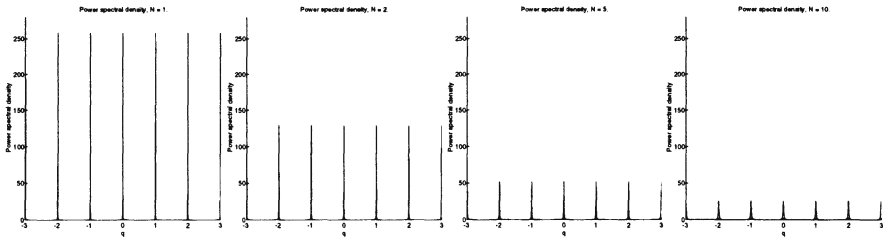


Figure 7 Power spectral density for two-state Markov chain source and dispersion factor $N = 1, 2, 5$ and 10 . The source is characterized by $p_{on}=1-6/1366$, $p_{off}=1-6/1802$.

For the one-stage multiplexer in Figure 2 (a), the mean queue size can be calculated. Assuming that the source consists of two independent identically distributed two-state Markov chains, and that the queue service rate is one, the mean queue size is, Li and Mark (1990)

$$\bar{q} = \frac{\bar{\epsilon}^2}{1 - 2\bar{\epsilon}} \cdot \left(\frac{2}{1 - \phi} - 1 \right), \text{ where } \bar{\epsilon} \text{ and } \phi \text{ are defined in (2) and (4).} \quad (12)$$

In case of cyclic dispersion, the dispersion may be illustrated by reducing the correlation from $\phi^{[n]}$ to $\phi^{N[n]}$, $N \geq 1$, according to (5), while keeping the peak and mean source rates unchanged. The mean queue size can then be expressed as

$$\bar{q} = \frac{\bar{\epsilon}^2}{1 - 2\bar{\epsilon}} \cdot \left(\frac{2}{1 - \phi^N} - 1 \right). \quad (13)$$

ϕ^N in (13) refers to the correlation sequence between adjacent packets from the same source, arriving at a queue. In the multiplexer model, however, the correlation behaves slightly different, since packets originate from different independent sources. The calculated and simulated values might therefore differ. The calculated mean queue size as a function of the dispersion factor is shown together with simulated results in Figure 8, and the results show accordance in behaviour between the calculated and simulated values.

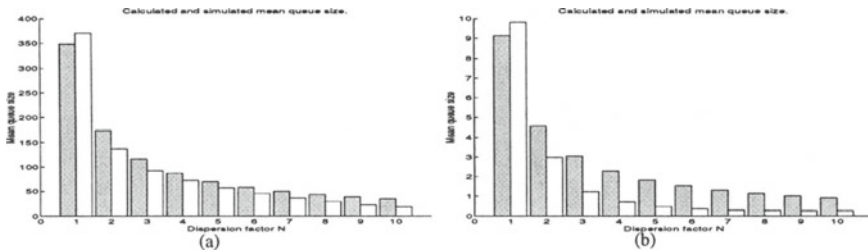


Figure 8 Calculated and simulated mean queue sizes, where each source consists of two Markov chains with $p_{on}=1-6/1366$, $p_{off}=1-6/1802$ (a) and $p_{on}=1-6/227$, $p_{off}=1-6/596$ (b). The shaded bars show the calculated mean queue size according to (13), and the utilization factor is 0.86.

3.3.2 Simulated queuing behaviour

Figure 9 shows the simulated mean queue size and standard deviation of the queue size, for N two-state Markov chain sources with peak rate 1, using the system model in Figure 2 (b). Both the mean and the deviation of the queue size decrease as N increases, but there is no obvious optimum N , as was the case for Poisson traffic. Nevertheless, one can see from the graphs that for $N=1..5$ there is a significant change in queue size, while both the mean and the deviation of the queue size start to level out for $N>5$.

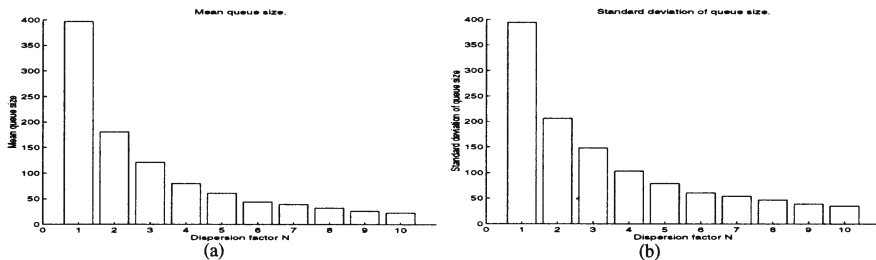


Figure 9 The mean queue size and standard deviation of the queue size for traffic generated by a two-state Markov chain source with peak rate 1, $p_{on}=1-6/1366$ and $p_{off}=1-6/1802$. The utilization factor is 0.86.

The sources used to obtain the results above produce strongly correlated traffic streams. In Figure 10, we investigate how the queuing behaviour is affected by the correlation. As the correlation decreases, an optimum dispersion factor shows up. Increasing the dispersion factor above that optimum value results in an increasing mean queue size. For highly correlated traffic, the mean queue size levels out at about $N=5$, and one may assume that for some larger N than shown in the graphs, the queue size starts to increase again. Additional results, which are not presented here, show that it is actually the case. Further results also show the standard deviation of the queue size to behave similarly to the mean queue size.

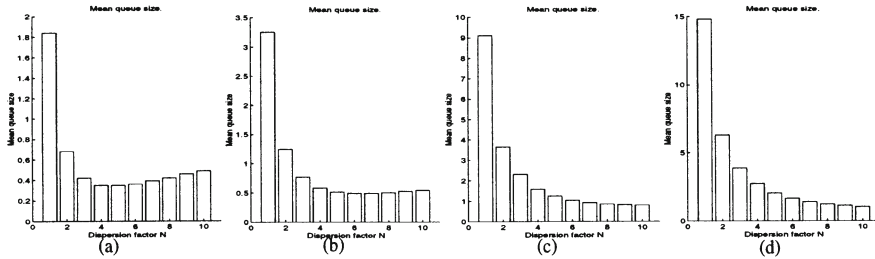


Figure 10 The mean queue size changing with the correlation. The traffic is generated by two-state Markov chains with $|\phi|=0.2$ (a), $|\phi|=0.4$ (b), $|\phi|=0.7$ (c) and $|\phi|=0.8$ (d). The source peak and mean rates are kept constant, and the queue utilization factor is 0.86.

Considering traffic with lower correlation, we also investigate how the optimum dispersion factor depends on the utilization factor in the queue. Figure 11 shows that the optimum dispersion factor increases as the utilization factor increases, so when increasing the utilization factor for low correlation traffic, the result approaches the ones obtained with high correlation.

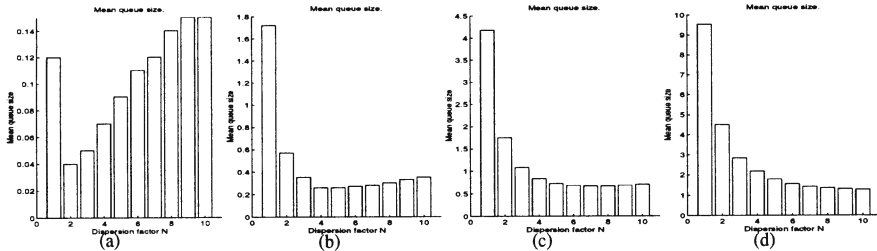


Figure 11 The mean queue size changing with the queue utilization factor. The traffic is generated by two-state Markov chains with $p_{on}=0.7$, $p_{off}=0.8$ and peak rate 1. The utilization factor in the different graphs is 0.48 (a), 0.72 (b), 0.84 (c) and 0.92 (d).

Next, we investigate how the queuing behaviour changes with dispersion for different source peak rates (Figure 12). Considering the case with no dispersion, that is, the first bar in each graph, we note that the mean queue size increases approximately proportionally to the source peak rate, when the utilization factor is constant. When looking at the graphs, we also note that dispersion improves the queuing behaviour more as the source peak rate increases. Additional results, which are not shown here, indicate a similar behaviour for the standard deviation of the queue size. The optimum dispersion factor still can be found in the interval [2, 5].

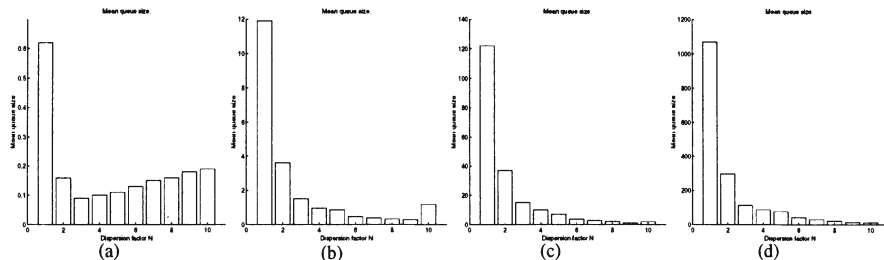


Figure 12 How the mean queue size changes with the source peak rate. The traffic is generated by two-state Markov chains with $p_{on}=0.7$ and $p_{off}=0.8$, and the queue utilization factor is 0.6. The peak rate is 1 (a), 10 (b), 100 (c) and 1000 (d).

Figure 13 gives an example of how the queue size varies in time for the system. Obviously, dispersion reduces the large fluctuations in queue size, hence making it easier to predict the amount of capacity needed for a source. Earlier results accordingly show a decrease in the equivalent capacity for dispersed traffic, Gustafsson and Karlsson (1996). For traffic generated by a two-state Markov chain, dispersion thus reduces the correlation in the traffic stream and thereby also the mean queue size and standard deviation of queue size. The more correlated traffic, and the higher queue utilization factor, the higher the gain obtained by using dispersion. For low correlation, low peak rate and low utilization factor, the dispersion factor should be rather small in order not to aggravate the queuing behaviour. In general, N should be in the interval $[2, 5]$. With $N > 5$, the surplus queuing benefits may not always justify the overhead caused by the required additional paths.

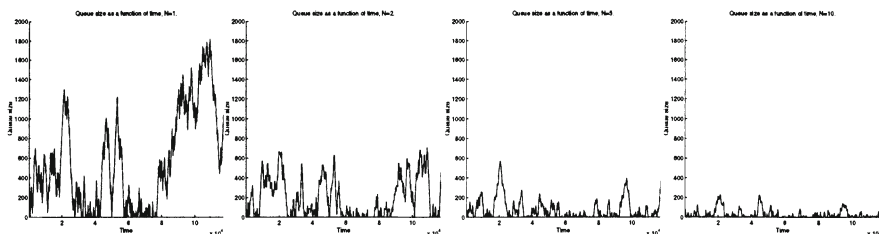


Figure 13 Example of queue size as a function of time for $N=1, 2, 5$ and 10 , for Markov chain sources with peak rate 1, $p_{on}=1-6/1366$, $p_{off}=1-6/1802$. The utilization factor is 0.86.

3.4 Queuing with traffic generated by the FPDl-map

3.4.1 Simulated queuing behaviour

For the values on the Hurst parameter considered here, the FPDl-map traffic source generates a queue length distribution with an unbounded mean queue size, Pruthi (1995). The unbounded queue size is due to the fact that the traffic is highly variable, wherefore a single burst occasionally may be large enough to dominate the queuing performance. We can thus not investigate how the mean queue size changes for different degrees of dispersion, as we did in Section 3.2 and Section 3.3. Instead, we look at the queue distribution (Figure 14). The results indicate that dispersion reduces the queue size for this type of traffic as well as for the traffic types discussed earlier. Figure 15 shows an example of how the queue size varies in time when dispersion is

used. These results indicate a similar change in queuing behaviour, due to traffic dispersion, as revealed earlier in this paper.

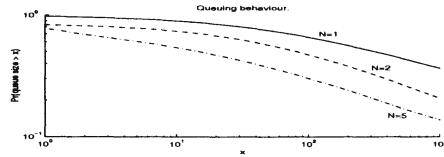


Figure 14 Queue-length distribution for traffic generated by the FPDl-map. The traffic has peak rate 1 and Hurst parameter 0.8333, and the utilization factor for the queue is 0.86. The graph shows results for $N=1, 2$ and 5 , and the axes show x and $p(x)=\Pr\{\text{queue size} > x\}$.

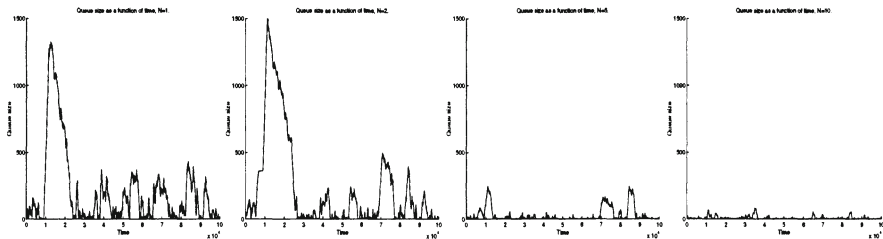


Figure 15 Example of queue size as a function of time for traffic generated by the FPDl-map, with peak rate 1 and Hurst parameter 0.8333. The utilization factor is 0.86, and the results are for $N=1, 2, 5$ and 10 .

3.5 Traffic dispersion - a general or individual decision?

As the previous results show, traffic dispersion may yield large performance improvements. An important remaining question is whether there are any benefits from using dispersion on one source alone, or if every user must employ dispersion to make the benefits show. We try to answer this question by again studying the model in Figure 2 (b), while adding non-dispersed background traffic to the traffic already generated into a queue. All sources are two-state Markov chains, and the sum of the peak rates of the dispersed source and the background traffic source is kept constant, while the dispersed traffic is varied from constituting 0 to 100% of the total traffic.

Figure 16 shows how the mean queue size changes with the amount of dispersed traffic. We note that in Figure 16 (a), the effects of dispersion show when about 20% or more of the traffic is dispersed, while the corresponding limit in Figure 16 (b) is between 20 and 30%. The sources in Figure 16 (b) are less correlated than the sources in Figure 16 (a), and we thus conclude that the stronger correlation in the traffic stream, the earlier the effects of dispersion start to show. In general, dispersion starts showing when employed on about 20% of the traffic, while the large benefits may come later. The results also further emphasize the statement that a dispersion factor larger than 5 does not significantly improve the performance.

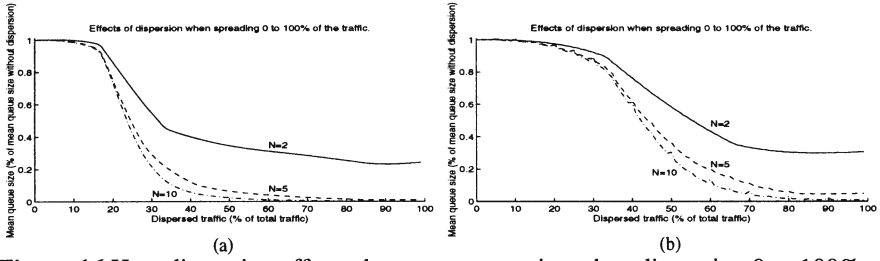


Figure 16 How dispersion affects the mean queue size when dispersing 0 to 100% of the traffic. The total peak rate of the arriving traffic is 100 packets/time unit. In (a), the sources are characterized by $p_{on}=0.7$, $p_{off}=0.8$ and the queue utilization factor is 0.6. The corresponding values in (b) are $1-6/1366$, $1-6/1802$ and 0.52 respectively.

4 OPTIMUM DISPERSION FACTOR

The results in the previous Sections show the optimum dispersion factor to be in the interval $N=[2, 5]$. The higher dispersion factor, the higher gain, but when N exceeds 5, the additional gain per new path starts to level out. Earlier results have shown the dispersion factor to depend on the ratio of the source peak rate to the link capacity, Gustafsson and Karlsson (1996). The higher ratio, the larger gain, and in general, dispersion is useful as the peak-to-link ratio exceeds about 1/10 or the peak-to-mean ratio exceeds about 10. Evidently, the dispersion factor for a call depends on the number of paths available in the network. As mentioned earlier, the paths should preferably be disjoint and of equal length. An effect of one path being longer or congested would be large resequencing delay at the receiver. The effects of a congested path could be avoided by forward error correction. If for example a single parity check is used, one of the paths would carry redundant information, hereby increasing the tolerance to link failures and information loss at the expense of higher capacity requirements. The effects of redundancy on the capacity can be illustrated by the effective bandwidth of a connection.

We consider the continuous-time version of the two-state Markov chain source from Section 2.2.2. With transition rates α and β from off to on and on to off state respectively, the effective bandwidth is given by, Kelly (1996)

$$a(s, t) = \frac{1}{st} \cdot \log \left\{ \left[\frac{\alpha}{\alpha + \beta} \quad \frac{\beta}{\alpha + \beta} \right] \exp \left(\begin{bmatrix} -\beta + hs & \beta \\ \alpha & -\alpha \end{bmatrix} t \right) \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}, \quad (14)$$

where s is the space-scale, relating to the expected quality, t is the time-scale and h is the source peak rate. With N paths, of which k carry dispersed information from the source, and $N-k$ carry redundant information, the effective bandwidth can be expressed as

$$a(s, t, N, k) = \frac{N}{st} \cdot \log \left\{ \left[\frac{\alpha}{\alpha + \beta} \quad \frac{\beta}{\alpha + \beta} \right] \exp \left(\begin{bmatrix} -\beta + sh/k & \beta \\ \alpha & -\alpha \end{bmatrix} t \right) \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}. \quad (15)$$

In order to better show the effects of dispersion on the effective bandwidth, we temporarily withdraw the restriction on N to be an integer. Some results are presented

in Figure 17. The graphs show that if $k=N-1$, a dispersion factor $N>3$ still requires less capacity than a non-dispersed connection. For $k=N-2$, this holds for $N>5$. Provided that there are five disjoint paths of equal length available in the network, a dispersion factor $N=5$ should be used, with or without redundancy, depending on the quality of service requirements from the user. If the number of available paths is less, a dispersion factor $N=2$ and $N=3$ gives a capacity gain without redundancy, while $N=4$ can tolerate redundant information to be carried on one of the paths. Lastly, it should be noted that the choice of dispersion factor may be different if there are special requirements, such as high security or extremely low capacity utilization on each path.

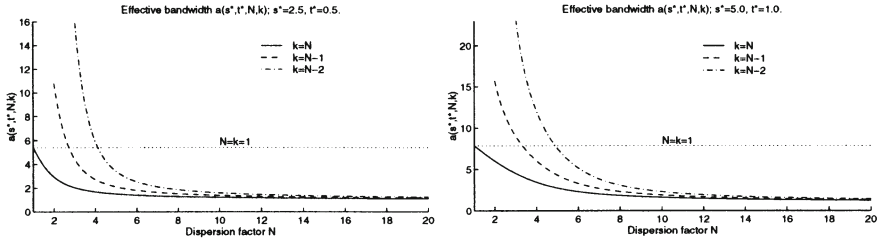


Figure 17 Effective bandwidth for a two-state Markov chain source, as a function of N and k . In the example presented, $\alpha=1$, $\beta=9$ and $h=10$. The s and t parameters are kept constant in each graph.

5 CONCLUSIONS

In this paper, we have investigated the basic properties of traffic dispersion. We have discussed how dispersion changes the traffic behaviour for different types of traffic sources, and how it affects the queuing performance. We confirmed that cyclic dispersion performs better than sequential dispersion, with respect to the queuing behaviour. When using cyclic dispersion, it is possible to reduce the correlation in the traffic stream and to reduce the peak rate of a source. The influence of a source on each of the paths used is hence reduced, and this makes it possible to fully utilize the benefits of statistical multiplexing in for example an ATM network. Reducing the correlation also gives a smoother queuing behaviour. Traffic dispersion has been shown to decrease the mean and variance of the queue size, and in a more general perspective to reduce the fluctuations in the queue size over time. The benefits from using dispersion are a reduction on the order of five to ten times in the mean and the standard deviation of the queue size, compared to non-dispersed transmissions. In order to make the effects of dispersion show though, the strategy must be employed on at least 20 to 30% of the total amount of multiplexed traffic. In summary, our results show the large potential of traffic dispersion, with an optimum dispersion factor $N=5$.

Considering the benefits from using traffic dispersion, the technique deserves further research and attention, with focus on studies of larger networks, delays through the networks and resequencing delays at the receiver. What makes the traffic dispersion approach special is that the network does not have to adapt to complicated traffic characteristics. On the contrary, the traffic characteristics are engineered to suit a reasonable network structure. This, we believe, gives us the opportunity of fast, secure and fault-tolerant transfers of data to satisfy high performance demands also for highly bursty traffic sources.

6 ACKNOWLEDGEMENTS

The simulations were performed on the sequential simulator YESS, developed by the Simulation Laboratory at KTH Department of Teleinformatics. The authors thank Robert Rönngren for his valuable help with the simulator and his interest in this work. We also thank Parag Pruthi at Bellcore, for providing us with - and explaining to us - the FPD1 map traffic generator.

7 REFERENCES

- Brady, P.T. (1968) A Statistical Analysis of On-Off Patterns in 16 Conversations. *The Bell System Technical Journal*, Vol. 47, 73-91.
- Gustafsson, E. and Karlsson, G. (1996) When Is Traffic Dispersion Useful? A Study On Equivalent Capacity, in *Performance Modelling and Evaluation of ATM Networks* (ed. D. Kouvatsos), Second volume, Chapman & Hall.
- Gustafsson, E. and Karlsson, G. (1997) A Literature Survey on Traffic Dispersion. To appear in *IEEE Network*, March 1997.
- Kelly, F. (1996) Notes on effective bandwidths, in *Stochastic Networks: Theory and Applications* (ed. F. Kelly, S. Zachary, I. Ziedins), Oxford University Press, 141-168.
- Kleinrock, L. (1975) *Queueing Systems, Volume I: Theory*, John Wiley & Sons.
- Lee, H.H. and Un, C.K. (1986) A Study of On-Off Characteristics of Conversational Speech. *IEEE Trans. on Communications*, Vol. COM-34, No. 6, 630-637.
- Leland, W.E. et al. (1994) On the Self-Similar Nature of Ethernet Traffic (Extended Version). *IEEE/ACM Trans. on Networking*, Vol. 2, No. 1, 1-15.
- Li, S-Q. (1989) Study of Information Loss in Packet Voice systems. *IEEE Trans. on Communications*, Vol. 37, No. 11, 1192-1202.
- Li, S-Q. and Mark, J.W. (1990) Traffic Characterization for Integrated Services Networks. *IEEE Trans. on Communications*, Vol. 38, No. 8, 1231-1243.
- Paxson, V. and Floyd, S. (1995) Wide Area Traffic: The Failure of Poisson Modeling. *IEEE/ACM Trans. on Networking*, Vol. 3, No. 3, 226-244.
- Pruthi, P. (1995) *An Application of Chaotic Maps to Packet Traffic Modeling*. Ph.D. Dissertation, TRITA-IT R 95:19, Royal Institute of Technology, Stockholm, Sweden.

8 BIOGRAPHIES

Eva Gustafsson received her M.Sc. degree in electrical engineering from the Royal Institute of Technology, KTH, in 1992. She is currently a Ph.D. student at the KTH Department of Teleinformatics, working on traffic dispersion in ATM networks.

Gunnar Karlsson received his master's degree from Chalmers University of Technology in 1983 and his Ph.D. from Columbia University in 1989. He has worked at IBM Zurich Research Laboratory from 1989 to 1992, and at the Swedish Institute of Computer Science since 1992. He is a member of the faculty at the Department of Teleinformatics at KTH, and is currently visiting professor at EPFL, Switzerland.

PART SEVEN

Bandwidth Allocation

Dynamic Bandwidth Allocation for Stored VBR Video in ATM End Systems

S. Gumbrich, H. Emgrunt, T. Braun
IBM European Networking Center
Vangerowstr. 18, 69115 Heidelberg Germany
e-mail: (gumbrich, braun)@heidelbg.ibm.com
Phone: +49 6221 594369, Fax: +49 6221 593300

Abstract:

The paper introduces a scheme for delivery of variable bit-rate (VBR) coded video from a video server over ATM networks. To solve the problem of long term variations in bandwidth requirements, a dynamic bandwidth allocation scheme is proposed. This is based on renegotiation of bandwidth during the duration of the call.

An algorithm is presented, that determines optimized renegotiation points for a given VBR coded video by minimizing cost functions. Appropriate cost functions, and methods for reducing the run-time expense of this optimizing algorithm are outlined.

Results are presented for real VBR MPEG video. The results show, that dynamic bandwidth allocation reduces the bandwidth requirements significantly and that the presented method leads to lower costs compared to other known renegotiation schemes.

Keywords

ATM, MPEG, bandwidth allocation, video transmission, VBR

1 INTRODUCTION

Compressed video traffic is expected to be a significant source of the traffic mix in asynchronous transfer mode (ATM) networks. There are two ways to compress video streams before transmitting them over networks: Constant Bit Rate (CBR) and Variable Bit Rate (VBR) compression. Note, that compressed video data is variable bit rate in nature.

When a video source is compressed using CBR techniques, the result is a constant bit rate stream with varying quality. At times of peak activity, quality decreases since there is not enough bandwidth to transmit all information. In contrast, VBR compression results in constant quality but varying bandwidth requirements. This constant quality is advantageous for the user but more difficult to support by the network. In this paper we address transmission of VBR coded video over ATM networks.

One important characteristic of compressed video is its burstiness over multiple time scales (Rathgeb, 1993),(Garret,1994). There is not only the short term variation in the source rate due to different frame-types and variations within a scene, but also a variation over long time scales due to scenes with different information contents. For example, in an MPEG compressed version of the “Star-Wars” movie, there are episodes with a sustained peak rate of five times the long term average rate (Grossglauser, 1995). Those episodes last for more than 10 seconds.

Short term variations can be smoothed by shaping the video stream (Lam, 1994),(Graf,1994). This is not possible for the long term variations due to restrictions of delay and buffer space. Moreover, the long term variations lead to a ATM traffic descriptor that is mainly based on the episodes with the highest bandwidth demands. Therefore, large amounts of mostly unused bandwidth have to be reserved for the whole movie. Since billing in the future will be based on the reserved bandwidth this leads to unnecessarily high costs.

To solve the problems of long term variations in transmission of videos we propose the concept of dynamic bandwidth allocation via renegotiation. Renegotiation means sending a signalling message along the path, requesting an increase or decrease of the current service rate corresponding to a new traffic descriptor. If the request is feasible, the network allows the renegotiation, and upon completion of the request, the source is free to send data compliant to the new traffic descriptor.

Our scheme is based on the ATM VBR service, extended by renegotiation functionality. The traffic contract for the VBR service includes an extended traffic descriptor. This traffic descriptor has to be determined by the user. A policing algorithm (generic cell rate algorithm, GCRA) checks the conformance of the arriving source traffic with this traffic descriptor (ATM Forum,1995). Non-Conforming cells are discarded or tagged indicating low priority. This has to be avoided since it leads to an uncontrollable service

degradation. On the other hand, the traffic descriptor should be as tight as possible to avoid over-reservation of bandwidth. Chapter 2 describes a way to obtain a deterministic traffic descriptor for stored video. Chapter 3 explains the concept of dynamic bandwidth allocation. The network model used for evaluation is described in chapter 4. In chapter 5 the algorithm for finding the renegotiation points and appropriate cost functions is presented. Schemes to accelerate the algorithm are discussed in chapter 6. Finally (chapter 7) we compare our algorithm to algorithms of other researchers.

2 DETERMINATION OF THE ATM VBR TRAFFIC DESCRIPTOR

The ATM Forum Traffic Contract (ATM Forum,1995) specifies the negotiated characteristics of an ATM Layer Connection at the UNI. It consists of a traffic descriptor and a set of QoS parameters. The traffic descriptor specifies the conforming cells of the ATM connection. It consists of the elements Peak Cell Rate (**PCR**), Sustainable Cell rate (**SCR**), Maximum Burst Size (**MBS**), Cell Delay Variation Tolerance (**CDVT**) and the Conformance Definition based on two GCRA algorithms.

The CDVT is independent of the source traffic and is therefore provided by the network adapter. It is the users task to determine the parameters PCR, SCR and MBS. An algorithm is necessary to select those parameters in such a way that no cells are tagged or discarded by the GCRA policing algorithm (i.e. to satisfy the conformance definition).

An *empirical envelope* is used (Cruz,1991), (Cruz,1995), (Knightly,1995b), (Wrege,1996) to give the most accurate, time-independent, deterministic characterization of the source traffic. We use the empirical envelope to determine the ATM traffic descriptor. The empirical envelope $\varepsilon(I)$ for an arrival function $b(t)$ is defined as shown in Figure 1. The quantity of data provided from the source within any interval of time I is less than a value $\varepsilon(I)$ that depends on the interval length.

For our experiments we assume that the traffic shaping function used is “frame period averaging” (Graf,1994). This results in equally spaced cells over each frame period (33ms). Usage of more advanced traffic shaping functions leads to lower values of the empirical envelope, but has no impact on the schemes presented here.

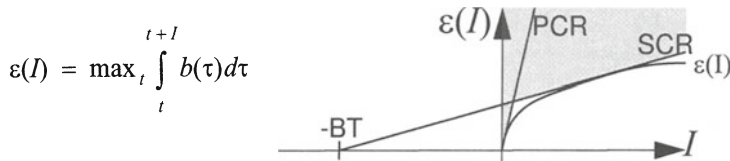


Figure 1: Empirical envelope and traffic descriptor

In Figure 1 it is shown how the source traffic descriptor is determined from a given empirical envelope. The PCR is determined by a tangent to the empirical envelope that

goes through the origin. On the contrary, there is an infinite set of {SCR,BT} pairs that lead to minimal conformant traffic descriptors. The only requirements for the construction of the SCR line are to touch and not to cross the empirical envelope. This makes sure that the traffic descriptor is conform to the source traffic and as tight as possible. After selecting one of the SCR values the BT is defined by the crossing of the SCR line with the x-axis. The sum of BT and the time for sending all cells of a connection specify the duration of that connection. This term is relevant for the definition of SCR. The signalling Parameter MBS is determined from

$$MBS = \left\lceil 2 + \frac{BT}{\frac{1}{SCR} - \frac{1}{PCR}} \right\rceil$$

The value 2 is added to ensure that the GCRA does not tag or discard cells conformant to the given empirical envelope $\varepsilon(i)$. This is necessary, due to the convention of the ATM Forum (ATM Forum, 1995) to use the following equation to derive BT_{GCRA} from the signalling parameters (MBS,SCR,PCR).

$$BT_{GCRA} = (MBS - 1) \times \left(\frac{1}{SCR} - \frac{1}{PCR} \right)$$

To select the value for SCR we use a heuristic scheme introduced in Wrege et al. (1996). In this scheme an additional criterion to minimize the weighted difference between the traffic descriptor and the empirical envelope is used. Note that this way of constructing the empirical envelope and the determination of the deterministic traffic descriptor only works for frame sizes known at connection setup time (i.e. stored video). This deterministic source traffic descriptor is mainly determined by long scenes with high bandwidth requirements. Since those scenes are common in real VBR MPEG videos, this leads to over reservation of bandwidth. Dynamic bandwidth allocation accounts for this behavior.

3 DYNAMIC BANDWIDTH ALLOCATION

Bandwidth allocation techniques for video can be divided into two classes: dynamic and static. If static bandwidth allocation is used, the traffic descriptor is valid for the complete duration of the connection. If the source traffic descriptor can be changed without setting up a new connection the term dynamic bandwidth allocation is used.

Dynamic bandwidth allocation is necessary since scenes with different information contents can result in extreme changes in the bandwidth requirements. If only static bandwidth allocation is performed, the worst case scenes determine the empirical envelope.

This is due to the fact that the empirical envelope is equivalent to the maximum number of data provided from the source for each interval within the validity of the source traffic descriptor. This leads to over reservation of bandwidth during parts (scenes) of the video with lower bandwidth requirements.

A requirement for dynamic bandwidth allocation is that the network allows renegotiation of the source traffic descriptor. ITU-T Q.2963 defines an extension of ITU-T Q.2931 signalling to allow renegotiation messages. The ATM Forum has no renegotiation messages defined (ATM Forum, 1996), but renegotiation is planned as future work item for the next release (Kaufman, 1996). With this knowledge we can expect that ATM networks will be capable of supporting renegotiation in the future.

During renegotiation a switch controller does not need to recompute routing, allocate a VCI or acquire housekeeping records. This reduces the renegotiation overhead. Nevertheless renegotiation is only feasible in time scales of several seconds (Lakshman, 1996) to avoid excessive signalling. The renegotiation frequency is a trade off between signalling overhead and a high utilization of the reserved bandwidth. A minimum of 1 second between two renegotiations and an average of more than 5 seconds is a good compromise.

The schemes dealing with video can be divided into two classes: off-line and on-line. For live video the frame sizes are not known at connection setup or renegotiation time. Schemes that deal with live video are called on-line schemes. In contrast, the frame sizes for stored video are known in advance. Schemes relying on stored video are called off-line schemes. There are two main problems that have to be solved for an off-line dynamic bandwidth allocation scheme. First, the positions have to be determined where the bandwidth requirements of the video change. An optimized solution for this topic is presented in this paper. Second, the bandwidth requirements of the video have to be adapted to the reserved traffic descriptor if the network denies the request for more bandwidth. We currently investigate this problem (Gumbrich, 1995).

Several scientists already addressed the problems of dynamic bandwidth allocation. Chong et al. (1995) discussed on-line schemes for CBR video. Their main interest is the prediction of the required bandwidth. Grossglauser et al (1995) developed on-line and off-line schemes for a CBR service. For off-line they defined constant costs per renegotiation and per bandwidth unit. This cost-function is then minimized to find the proper renegotiation points. This optimizing algorithm is intended for renegotiated CBR service. We compare this algorithm to our algorithm in chapter 7.2 of this paper. For on-line they predict the next frame size and start renegotiation depending on the predicted frame size and the CBR buffer level.

Renegotiation for on-line VBR video is addressed by Pancha et al. (1993) and Knightly et al. (1995). Pancha et al. rely on the Fast Reservation Protocol and allows renegotia-

tion for each new frame. Their prediction is based on cell statistics. Knightly et al. expect the network to use the D-BIND model (Knightly, 1994) as traffic descriptor. The ATM Forum Traffic descriptor is a subset of the D-BIND model. On-line and off-line schemes are addressed in their work. Their off-line algorithm for VBR video is very efficient but not optimal. We compare this algorithm to ours in chapter 7.1

(Reininger, 1995), (Bansal, 1995), (Mark, 1996) concentrate on determining VBR traffic descriptors for on-line video. They come up with a complete system that is capable of determining on-line renegotiation points. This system increases the quantization values of the coder, if the renegotiation request is not granted by the network. El-Henaoui et al. (1996) investigate a renegotiation system for on-line MPEG 1 audio. Their system is based on fast reservation. Lakshman et al. (1996) implemented and evaluated a renegotiation system with FORE switches and SUN workstations. They use the ATM CBR service (PCR reservation) for transmission of VBR videos.

Our concept is different to others in some important points. A renegotiation scheme for off-line VBR video, based on the ATM VBR service is investigated. We present the only optimizing algorithm for finding renegotiation points in this scenario. A further advantage of our work is, that we adjust the minimal distance between two renegotiations and the number of renegotiations in a video by a cost function for renegotiations. This is necessary to avoid overloading the signalling system of the network.

4 NETWORK MODEL

We need a model for the evaluation of bandwidth requirements. This is straight forward for CBR traffic since there is only one numerical value the PCR. For VBR the set {PCR, SCR, BT} has to be mapped to one value with a strict order to be comparable. This value is the maximum number of deterministic connections that can be transmitted by a given switch. Therefore, we use the model in Figure 2..

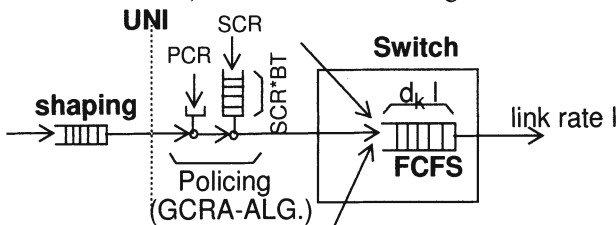


Figure 2: network model

In this model the switch uses a simplified **deterministic** call admission control (**CAC**) algorithm based on worst case assumptions. This allows for deterministic QoS guarantees, but leads to a lower network utilization compared to a statistical model. The prob-

lem with statistical models is, that it is still unsolved how to use the deterministic ATM source traffic descriptor to efficiently build a statistical descriptor for the CAC. The switch in Figure 2 has output buffers for multiplexing different incoming connections to outgoing links. We assume here a First-Come-First-Served scheduling (FCFS) strategy for the output buffer. This means that all packets are transmitted in the order of their arrival. The decision to use more advanced strategy like earliest deadline first or static priority (Knightly,1994), (Knightly,1995b), (Liebeherr,1994), (Cruz,1991) has no impact on our investigations.

Under the worst case assumption that all videos are identical and transmitted synchronously the conditions to guarantee a given delay bound d_k for each connection can be expressed as:

$$\{ \max_{noc} | (noc \cdot b(t) - l \cdot t) \leq d_k \cdot l \} \quad \forall t$$

In this formula the link speed is denoted as l , $b(t)$ is the source traffic descriptor and **noc** is the maximum number of connections that can be supported. Note, that the **noc** value is low if the bandwidth requirement of a given video stream is high. We use the **noc** value to determine the evaluation function and the cost functions.

An arbitrary shaping method can be chosen for shaping. For our investigations we assigned the method frame period averaging (Graf,1994). At the network edge a policing algorithm (GCRA) checks the conformance of incoming cells to the source traffic descriptor.

The evaluation function $B(S)$ and the cost functions are defined based on this network model. $B(S)$ is used for estimation of bandwidth requirements throughout this paper. The area between two renegotiations is called segment. Therefore, the off-line algorithm divides the video into the segments $\{S_1, \dots, S_m\} = S$. With this the evaluation function $B(S)$ is defined as:

$$B(S) = \sum_{i=1}^m \frac{noc_i \times L_i}{L_S}$$

noc_i = number of connections for segment S_i

L_i = length of segment S_i

L_S = length of video

$B(S)$ is equivalent to the weighted average of the number of admissible connections introduced in Knightly et al. (1995). With the knowledge of the long term average rate and the link speed this is converted to the average utilization in (Knightly, 1995).

5 OPTIMIZING OFF-LINE ALGORITHM O-RVBR

The goal of the off-line algorithm presented here is to find the optimal renegotiation points. Therefore we basically check all possible renegotiation points and decide which ones are associated with the lowest costs. Costs are considered for the amount of reserved VBR-bandwidth (source traffic descriptor) and the renegotiation itself. The cost function $K(S)$ is defined as

$$K(S) = \left(\sum_{i=1}^{m-1} k(S_i) + r(S_i) \right) + k(S_m)$$

$k(S_i)$ = bandwidth cost for segment S_i
 $r(S_i)$ = Cost for renegotiation at end of Segment S_i
 m = number of segments

The goal of the algorithm is, to find the set of segments $S = \{S_1, \dots, S_m\}$ for which the cost function $K(S)$ is minimized. We call the algorithm **O-RVBR** ("Optimizing algorithm for Renegotiated VBR video").

Since off-line algorithms rely on stored video, the frame sizes are known in advance. The determination of the renegotiation points and the corresponding traffic descriptors for each segment has to be done only once. As the results are stored together with the video at the video server, it is not necessary to do the calculations in real-time.

5.1 Bandwidth costs

ATM VBR traffic will be billed, depending on the amount of reserved bandwidth. Otherwise users could reserve huge amounts of unused bandwidth without paying for it. Therefore we considered the following aspect. For the network operator it is most advantageous to maximize the number of possible connections (with their requested QoS) for a given link speed. This reduces the cost per connection. We investigated different cost functions (Emgrunt, 1996) based on the noc_i value defined in chapter 4. The most suitable function is

$$k(S_i) = \frac{l}{noc_i} \times L_i$$

l = link rate
 L_i = Length of Segment S_i

The term l/noc_i is the amount of bandwidth that is reserved for Segment S_i in the CAC. This mainly determines the bandwidth cost. The term L_i avoids that cuts are detected at points with constant bandwidth requirement ($k(S_i)$ does not change). We proved [Emgr96] that L_i is necessary even if the renegotiation costs are decreasing linearly or exponentially with the length of the segment.

5.2 Renegotiation costs

The least bandwidth is reserved (lowest bandwidth cost) if renegotiation is performed after each frame. This is not suitable for common ATM switches since this will lead to an extraordinary high signaling overhead. Therefore, we introduce a cost function for renegotiation. In (Lakshman, 1996) it is shown by measurement that renegotiation in

the range of “every few seconds” is suitable for ATM switches. To avoid unacceptable signaling overhead we introduced a minimal time interval MINTIME between two renegotiations. We use MINTIME=1sec in our investigations.

Ingenious renegotiation cost functions are constant or decrease with increasing segment length. We compared exponential with constant cost functions and drew the conclusion that with the same number of renegotiations the exponential cost function always leads to a lower rating value $B(S)$ if the exponent is significantly larger than zero. Therefore only constant cost functions are investigated further. There are basically 3 possible ways to determine the constant value $r(S_i)=\gamma$. This value is responsible for the resulting number of segments.

- stream independent: This is the easiest way to determine γ . It is useful if there are real costs per renegotiation and for VBR bandwidth specified by the network providers. This also assumes a network service, that only restricts the number of renegotiations by the value of MINTIME.
- stream dependent: With this method the renegotiation costs depend on statistical characteristics of the video stream. Video streams with high variance in bandwidth requirements are assigned higher values for γ . The advantage of this method is, that the load on the signalling system of a switch increases slowly with increasing variance of bandwidth requirements.

$$r(S_i) = \begin{cases} \infty & , \forall (\Delta t < MINTIME) \\ \frac{\sigma}{\mu} \cdot noc_{all} \cdot \alpha & , \forall (\Delta t \geq MINTIME) \end{cases}$$

In this formula σ is the variance and μ is the mean of the frame sizes of the video. σ/μ is a measure for the burstiness of the video stream. The number of connections if no renegotiation takes place is noc_{all} . The weight α determines how aggressive the algorithm segments the video stream. Our experiments showed that $\alpha=70$ is a good trade-off between number of renegotiations and bandwidth requirements.

- fix number of renegotiations: With this method a fix number of renegotiations is allowed for a movie. This is very useful for comparing different schemes. Moreover in practical applications this makes sure that the signalling system of the switches is not overloaded. Our algorithm to calculate the renegotiation costs is based on approximations.

5.3 Optimizing algorithm O-RVBR

The algorithm to find the optimal renegotiation points is presented in Figure 3. The input for this algorithm is the list of frame sizes of the actual video. The output is a list of optimal renegotiation points.

The algorithm computes the traffic descriptor for the current segment S_i (i.e. the interval of frame i to frame $i+\text{length}$). Then, the bandwidth costs $k(S_i)$ and the renegotiation costs $r(S_i)$ for this segment are calculated. The costs for each segment plus the costs of the remaining segments up to the last frame are calculated. If this costs are the lowest for the current starting point, they are termed the minimal costs ($\text{cost}(i)$). This test is done for all possible segment sizes beginning from the current starting point (i). Then, the current starting point is moved. Note that this algorithm starts from the last frame and moves the current starting point into the direction of the first frame. The result of the algorithm is a chain of optimized renegotiation points for each start point ($\text{segment_len}()$).

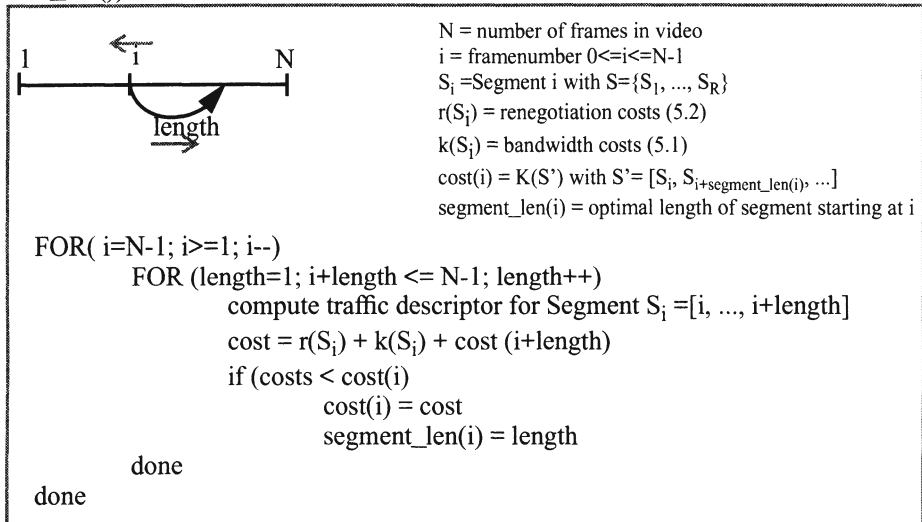


Figure 3: optimal off-line algorithm

In Figure 4 we show the rating $B(S)$ for the video trailer depending on the number of segments (number of renegotiations + 1) and the delay bound d_k . Increasing the number of renegotiations results in higher values of $B(S)$. This is valid for all different delay bounds d_k . From this it gets obvious that with a given QoS requirement (d_k fix) a system without renegotiation (1 segment) reserves always more bandwidth (higher costs, $B(S)$ lower) than a system with our renegotiation scheme. Moreover, it gets obvious that a high number of renegotiations is always superior to a lower number. The allowed num-

ber of renegotiations is unfortunately limited by the signalling system of the ATM switches [LakYa96]. To adjust the number of renegotiations our renegotiation cost functions $r(S)$ are provided (chapter 5.2).

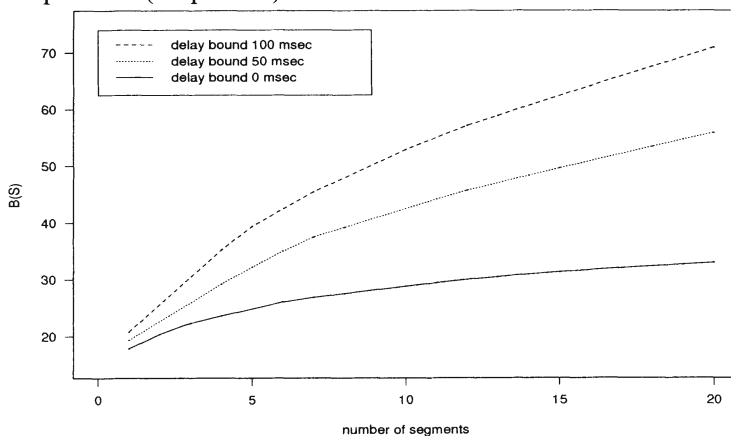


Figure 4: impact of numbers of renegotiation on $B(S)$

6 ACCELERATING THE ALGORITHM

The expense of the O-RVBR algorithm is $O(n^3)$ (Emgrunt, 1996), which is a lot if movies with $n = 162000$ frames (90 minutes) are taken into consideration. There are several possibilities for speeding up the algorithm with little impact on the optimized points for renegotiation. We discuss the three schemes empirical envelope extrapolation, short time noc, mean estimation in the next subsections.

The scheme “short time noc” proved to be the most advantageous acceleration scheme since it lead to the highest run-time reduction and showed the lowest impact on the evaluation function $B(S)$. We show that the combination of all three schemes further decreases the run-time.

6.1 Empirical envelope extrapolation

The operation with the highest expense in the inner loop of the optimal algorithm is the calculation of the empirical envelope. (This is part of computing the Traffic descriptor in Figure 3.) The expense for calculating the empirical envelope is $O(L_i^2)$ (L_i = segment length). By keeping the results from the last iteration step and checking only the difference introduced by the new frame the expense for calculating the empirical envelope is reduced to $O(L_i)$. This doesn't lead to any deviation from the optimal values and is therefore part of the optimal algorithm with expense $O(n^3)$.

The empirical envelope extrapolation is a scheme presented by Wrege (1996). The basic

idea is to calculate the empirical envelope for a maximum interval length of k with $k \leq L_i$. This reduces the expense from $O(L_i)$ to $O(k)$. To get a complete envelope those first k parameters are repeated L_i/k times. Since the empirical envelope is subadditive this leads to a valid traffic constraint function. This traffic constraint function is not as tight as the empirical envelope which results in lower noc values and lower values of the evaluation function $B(S)$. The expense of the algorithm is reduced to $O(k n^2)$ with Empirical envelope extrapolation.

6.2 Short time noc (k - noc)

This scheme basically reduces the number of runs through the inner loop by changing the end condition ($N-1$).

Our investigations showed that similar noc values for neighboring start points of segments in many cases result in the same endpoint for the optimal segment. Further it is obvious that the part of a segment with the highest bandwidth requirements is mainly responsible for the noc -value of the whole segment.

Based on our observations we developed a scheme called short-time noc . Short time noc calculates the empirical envelope from a given start point “ a ” of a segment to “ $a+k$ ” (i.e. for the frames $[a...a+k]$). On the basis of this empirical envelope we determine the traffic descriptor and the noc value for this segment. This noc value is called $knoc_a$. The algorithm compares the value $knoc_a$ with the value $knoc_{a+1}$ and derives thereof the end condition of the inner loop. We distinguish three cases (Figure 5):

1. $knoc_a = knoc_{a+1}$: the segments are very similar in this case. Therefore the end condition of the inner loop is set to the optimal end e of the segment starting with $a+1$.
2. $knoc_a > knoc_{a+1}$: extending $[a...a+k]$ to $[a...a+k+1]$ will reduce $knoc_a$ to $knoc_{a+1}$. Therefore optimal endpoints won't differ much and the end condition is set identical to case 1.
3. $knoc_a = knoc_{a+1}$: in this case the next segment $[i...i+k]$ ($i > a$) with $knoc_i < knoc_a$ is searched and the end condition of the inner loop is set to the optimal end of the segment starting with i .

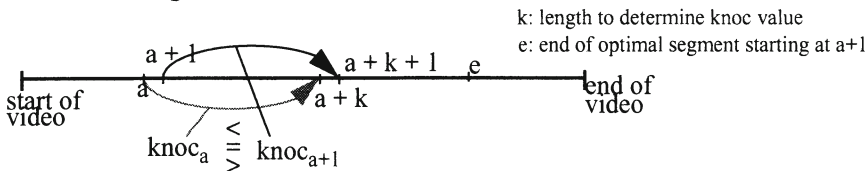


Figure 5: comparison of neighboring $knoc$ values

6.3 Mean estimation

In contrast to the short time noc scheme this scheme accelerates the algorithm by completely skipping the inner loop if suitable. This is our only scheme that takes into account the frametype characteristics of MPEG streams. Note, that the other schemes work also for non MPEG VBR video streams.

Mean estimation calculates the mean of the frame sizes for each frametype (I,P,B) of the segment $[a+1...e]$. Frame a is compared to the mean of frames of the same type. If the difference of frame a 's size to the mean is smaller than a threshold the segment $[a+1...e]$ is extended to $[a...e]$. This is repeated with frames $a-1$, $a-2$, ... until the threshold is exceeded. Only in this case the inner loop is processed.

6.4 Results

We show in table 1 a run-time example for the O-RVBR algorithm with the different acceleration schemes. We used a MPEG2 coded version of the trailer from the video Terminator 2. The length of this video is 1800 frames (1 minute). The cost functions were chosen identical for all schemes and the computation was done at a PC with a 133MHZ Pentium processor under Linux 2.0.

Table 1: run-time for Terminator 2 trailer

	<i>O-RVBR without acceleration</i>	<i>Empirical envelope extrapolation</i>	<i>short time noc</i>	<i>mean estimation</i>
run-time	4h 23min	1h 34 min.	23 min.	1h 17 min.
expense	$O(n^3)$	$O(k n^2)$	$O(n^3)$	$O(n^3)$
B(S)	57,08	52,85	57,08	52,53
parameters		$k=300$	$k=180$	$(I,P,B)=(10,20,40)$ cells

The results show that short time noc leads by far to the lowest run-time. At the same time short time noc has the highest values of the evaluation function B(S). This means that the renegotiation points found by this scheme are identical to the values found by O-RVBR. With appropriate parameters we can increase the evaluation function although for the other schemes, but this leads to an increased run-time. The value B(S) for k-noc is only decreased if the value for k is chosen extremely small ($k = 30 \Rightarrow B(S)=56,25$). Therefore k-noc is a good choice to reduce run-time. But keep in mind that reductions of B(S) are possible for certain videos.

To further reduce the run-time of the O-RVBR algorithm (especially for complete 90 minute videos) the acceleration schemes can be combined. Table 2 shows results for

combinations of different schemes with the same parameters as in Table 1.

Table 2: combination of different schemes for video trailer

	<i>Emp. Env. extr.</i> + <i>mean estim.</i>	<i>short time noc</i> + <i>mean estim.</i>	<i>short time noc</i> + <i>Emp.Env. extr.</i>	<i>Combination</i> <i>of all three</i> <i>schemes</i>
run-time	43 min.	18 min.	13 min.	11 min.
B(S)	53,63	52,62	53,48	53,48

Considering the data of table 2 it becomes obvious that by combining the three schemes additional run-time savings are possible. The evaluation function B(S) shows results that are comparable to the single schemes mean estimation and empirical envelope extrapolation. But the combination leads to a much lower run-time, than for the single schemes mean estimation and empirical envelope extrapolation. If we consider run-time and B(S), the best results are achieved by the combination of all three schemes.

We conclude that the best choice is short time noc if the run-time of the optimal algorithm has to be reduced. If further reduction is necessary a combination of all three schemes is the best choice. For videos that are likely to be played very often we propose to go for the highest possible value of B(S) while the run-time reduction of the algorithm may be more convenient for videos that are requested not to often from the video server.

7 COMPARISON TO WELL KNOWN SCHEMES

This chapter compares our algorithm with the VBR off line algorithm for the DBIND model presented by Knightly et al. (1995) and with the CBR off-line algorithm RCBP of Grossglauser et al. (1995).

7.1 DBIND off-line Algorithm

The D-BIND model is a general traffic model developed by E.Knightly and H.Zhang (1994). We use their algorithm with the ATM traffic descriptor to make it comparable to our less general approach. We show that for this special usage with the ATM VBR service our algorithm leads to better results.

The DBIND off-line algorithm (Knightly, 1995) detects the renegotiation points much faster than our algorithm. The expense of this algorithm is $O(n \log n)$ compared to $O(n^3)$ or $O(k n^2)$ for our algorithm. On the other hand the renegotiation points of our algorithm lead to higher values of B(S) which is associated to reserving less bandwidth. Since the off-line algorithm is executed only once for a video, and the resulting renegotiation

points are stored with the video on the video server our approach is advantageous for this class of applications.

The D-Bind algorithm finds the worst case part of the video that has the highest bandwidth requirement (Worst Case D-BIND parameter R_n). This part is increased to the right an left until the mean bandwidth of this part is lower than a threshold. This part is then termed a segment. The same procedure is repeated with the remaining parts of the video left and right of the segment.

For the DBIND algorithm we used the parameters $I_n=1$ second and ψ is varied to get different numbers of segments. The tests were done with different videos. "Trailer" is a MPEG 2 video we captured on our own. The other videos are MPEG 1 traces kindly provided by the University of Würzburg (<ftp-info3.informatik.uni-wuerzburg.de>). Figure 6 and Figure 7 show that O-RVBR is superior to the D-BIND algorithm independent of the number of renegotiations (number of segments -1) and the delay bound d_k in the ATM switch.

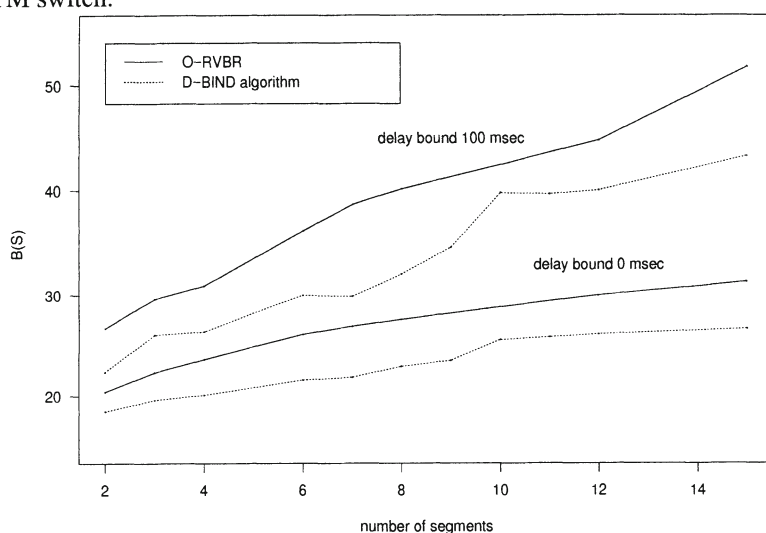


Figure 6: comparison of D-Bind and O-RVBR algorithm for video **trailer**

Figure 8 compares the O-RVBR algorithm and the D-Bind algorithm with a periodic scheme. The periodic scheme divides the video sequence into segments of equal size (length of segment = L_s / number of segments). It does not take the behavior of the video stream into consideration. In this comparison the value of $B(S)$ is displayed for different delay bounds. The number of segments is fixed to 12. Here it gets obvious that the D-Bind algorithm does only little better than the periodic scheme and that O-RVBR is always better.

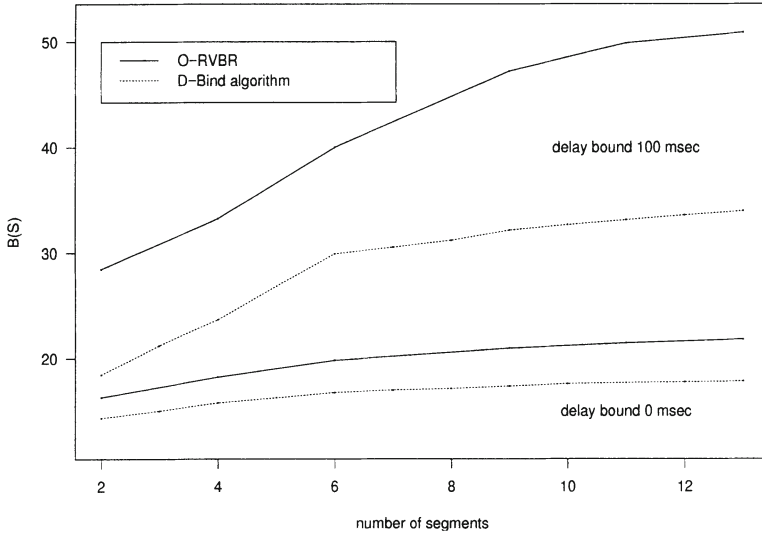


Figure 7: comparison of D-Bind and O-RVBR algorithm for video *asterix*.

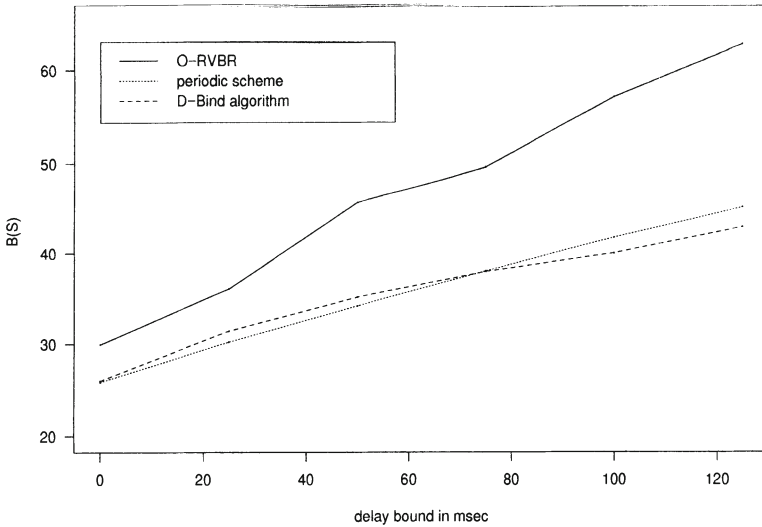


Figure 8: Comparison with periodic scheme for video *trailer*

7.2 RCBR algorithm

The off-line RCBR algorithm presented by Grossglauser et al. (1995) reserves a constant rate (PCR) for each segment. This is done under the requirement that there is no

loss in a given smoothing buffer B . This results in transmission of VBR video using a renegotiated CBR ATM service (piecewise constant rate reservation).

Our goal is to compare the two algorithms by excluding the problems of comparing CBR versus VBR (Quality, delay, statistical multiplexing). Therefore, we chose the following approach.

- We use the same delay introduced by RCBR's smoothing buffer B for smoothing the input of our O-RVBR algorithm. We developed a shaping scheme for a renegotiation service. This shaping scheme is used instead of frame period averaging (chapter 4) for the comparison with the RCBR algorithm.
- The delay bound d_k equals zero to disable the multiplexing functionality for the VBR service. With $d_k = 0$ only the PCR value is relevant for the traffic descriptor. Since SCR and BT have no impact the calculation of noc_i is reduced to $noc_i = \text{link rate} / \text{PCR}_i$

With this approach we compare the transmission of videos with equal quality and delay (QoS). The main difference are the algorithms for finding the renegotiation points. It has to be noted, that the run-time expense and the memory expense of the RCBR algorithm is $O(K*B*N)$ with K =number of bandwidth levels, B =buffer size and N =length of video. The run-time expense is much lower than for our algorithm, but the memory expense may be critical since RCBR leads to memory demands of 10MByte for short examples with $N=2000$ frames (66 seconds), $B=200$ cells and $K=20$ levels.

For our investigations we use the same videos as for the comparison with the DBIND algorithm. The number of bandwidth levels is chosen to be $K=20$ as recommended in (Grossglauser, 1995).

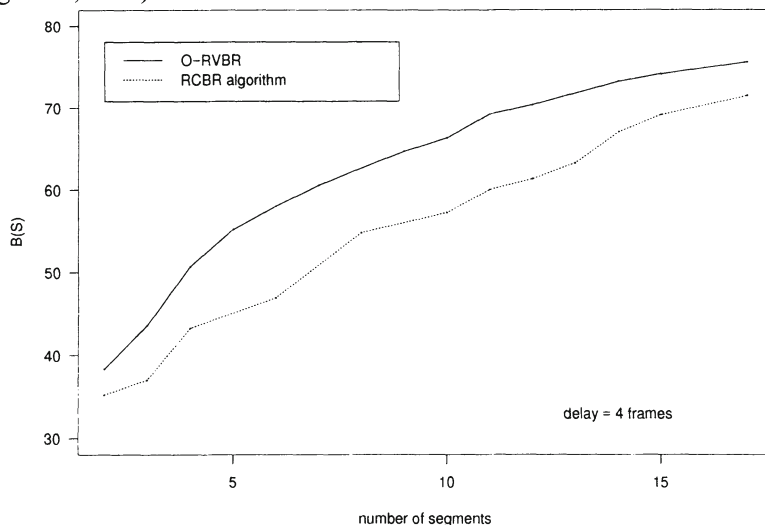


Figure 9: comparison of O-RVBR and RCBR for video trailer

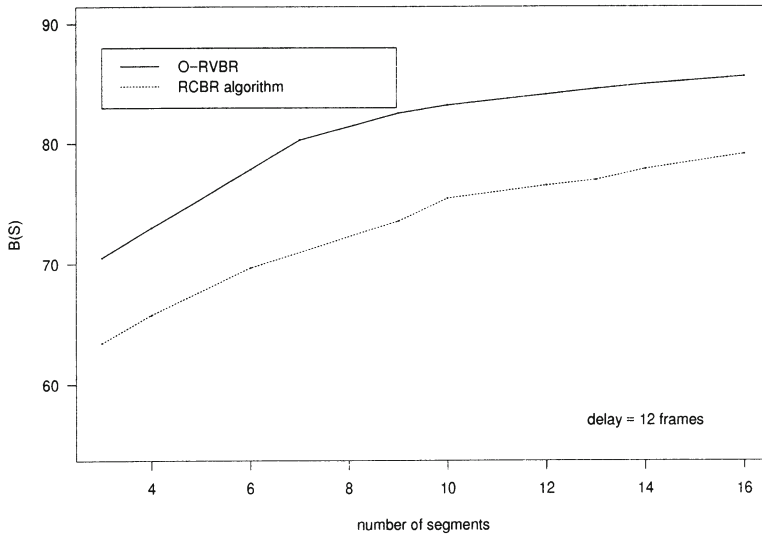


Figure 10: comparison of O-RVBR and RCBR for video **asterix**

We use different delays (4, 12 frame times) for the examples in Figure 9 and Figure 10. From these figures it gets obvious that O-RVBR leads to higher values of $B(S)$ than the RCBR algorithm. The penalty for this advantage is the increased run-time. Again, the run-time is not very critical from our point of view, since the algorithm is started only once for each video. It stores the optimized renegotiation points together with the video on the sever. Moreover the RCBR algorithm is designed especially for CBR, and is not applicable for VBR services since all different combinations of PCR, SCR and BT are possible. This would lead to an unacceptable high value of bandwidth levels K .

8 CONCLUSION

In this paper, we presented a scheme for transmitting stored VBR video over ATM networks. The scheme is based on dynamic bandwidth allocation. We showed for a deterministic CAC scheme that dynamic bandwidth allocation increases the number of possible connections for a given link rate and delay bound considerably. This is made possible by reducing the inevitable over-reservation of static bandwidth allocation schemes.

We developed an algorithm (O-RCBR) that determines the optimized renegotiation points. This is done by minimizing a cost function based on costs for bandwidth and costs for renegotiations. Appropriate cost functions for a VBR service are presented. Since the optimizing algorithm results in high run-time expense we developed accelera-

tion schemes. Especially the scheme k-noc proved to reduce the run-time considerably, while finding the same renegotiation points as the optimal scheme in most cases. The comparison of O-RVBR with existing off-line algorithms (Grossglauser, 1995), (Knightly, 1995) proved that O-RVBR leads to the lowest amount of reserved bandwidth. This advantage is obtained by using an algorithm with higher run-time. This is tolerable for the addressed application of a video server, since the algorithm runs only once per video. The resulting optimized renegotiation points are stored together with the video on the server.

9 REFERENCES

- ATM Forum (1995), ATM User-Network Interface (UNI) Specification Version 3.1,
 ATM Forum (1996), Traffic Management Specification Version 4.0, February 96
 Bansal V., Siracusa R., Hearn J., Ramamurthy G., Raychaudhuri D. (1995), Adaptive
 QoS-Based API for ATM Networking, NOSSDAV 1995, Durham, New Hampshire
 Chong S., Li S., Ghosh J. (1995), Predictive Dynamic Bandwidth allocation for Efficient
 Transport of Real Time VBR Video over ATM, IEEE JSAC, Vol 13, No1
 Cruz R.L. (1991), A Calculus for Network Delay, Part I and II, IEEE Transactions on
 Information Theory, Vol37, No.1, January 1991
 Cruz R.L. (1995), Quality of Service Guarantees in Virtual Circuit Switched Networks,
 IEEE Journal on Selected Areas in Communications Vol.13, No.6, August 95
 El-Henaoui S., Coelho R., Tohme S. (1996), A Bandwidth Allocation Protocol for
 MPEG VBR Traffic in ATM Networks, Infocom 96, San Francisco
 Emgrunt H. (1996), Dynamic bandwidth reservation for transmission of stored MPEG2
 video, Diploma thesis, University of Karlsruhe, September 1996
 Garret M.W., Willinger W. (1994), Analysis Modelling and Generation of Self Similar
 VBR Video Traffic, ACM SigComm, London, September 1994
 Graf M. (1994), Traffic Shaping for VBR Video, Proceeding 4th Open Workshop on
 High Speed Networks, Telecom Bretagne, Brest France
 Grossglauser M., Keshav S., Tse D. (1995), RCBP: A Simple and Efficient Service for
 Multiple Time-Scale Traffic, ACM SIGCOMM 95
 Gumbrich S., Sandvoss J., Comparison of Media Scaling Techniques for Real Time
 Packet Video Transport, Proc. of IEEE HPCS'95, Mystic Connecticut
 ITU-T (1993) Recommendation Q.2931, B-ISDN Signalling System
 ITU-T (1995), Recommendation Q.2963 (DRAFT), B-ISDN Connection Modification
 Kaufman J. (1996), Status of ATM Forum Technical Work Groups, January, 1996

- Knightly E., Zhang H.(1994), Traffic Characterization and Switch Utilization using a Det. Bounding Interval Dependent Traffic Model, U.C. Berkeley, TR-94-047
- Knightly E., Zhang H.(1995), A New Approach to Support Delay-Sensitive VBR Video in Packet-Switched Networks, NOSSDAV 1995, Durham, New Hampshire
- Knightly E., Wrege D., Liebeherr J.,Zhang H.(1995), Fundamental Limits and Tradeoffs of providing det. guarantees to VBR Video traffic, ACM SIGMETRICS'95
- LakshmanK., Yavatkar R.(1996), An Empirical Evaluation of Adaptive QoS Renegotiation in an ATM Network, NOSSDAV 1996, Zushi, Japan
- Lam S., Chow S., Yau D. (1994), An Algorithm for Lossless Smoothing of MPEG Video, SIGCOMM 94, London England
- Liebeherr J.,Wrege D.,Ferrari D.(1994), Exact Admission Control for Networks with bounded Delay Service, University of Virginia, Technical Report CS-94-92
- Mark B., Ramamurthy G.(1996), Real Time Estimation and Dynamic Renegotiation of UPC Paramaetrs for Arbitrary Traffic Sources over ATM, 7th International Workshop on Packet Video, Victoria, Australia
- Pancha P., Zarki M.(1993), Bandwidth Requirements of Variable Bit Rate MPEG Sources in ATM Networks, Modelling and Performance Evaluation of ATM Technology, ,Elsevier Science Publishers
- Rathgeb E. (1993), Policing of realistic VBR Video Traffic in an ATM network, International Journal of Digital and analog Communication systems. Vol. 6, 213-226, 1993
- ReiningerD., Ramamurthy G., Raychaudhuri D.(1995), VBR MPEG Video Coding with Dynamic Bandwith Renegotiation, IEEE International Conference on Comm.
- Wrege D.,Liebherr J.(1996), Video Traffic Characterization for Multimedia Networks with Deterministic Service, IEEE Infocom 96

10 BIOGRAPHY

Stefan Gumblich received his diploma degree in Electrical and Computer Engineering from the Fachhochschule Bingen and the University of Tennessee, Knoxville in 1991. In 1994 he received his diploma degree in Computer Science from the University of Paderborn. Since 1994 he has been a guest scientist at IBM European Networking Center in Heidelberg (Germany). His main research interests are the aspects of video transmission over high speed networks and the design and implementation of high speed network protocols.

Heinrich Emgrunt received his Diplom in Computer Science from the University of Karlsruhe in 1996. The subject of his diploma thesis was the transmission of stored video over ATM-networks.

Torsten Braun received his diploma degree and his doctoral degree in computer science from the University of Karlsruhe (Germany) in 1990 and 1993, respectively. From 1990 to 1994 he was a research assistant at the Institute of Telematics, University of Karlsruhe. From 1994 to 1995 he was a visiting scientist at the Institut National de Recherche en Informatique et en Automatique (INRIA) in Sophia-Antipolis (France). Since September 1995, he has been a guest scientist at the IBM European Networking Center (ENC) in Heidelberg (Germany). His main research interests are the implementation of transport systems for multimedia communication and the integration of Internet protocols and ATM.

Analysis of An End-to-End Proportional Bandwidth Allocation Algorithm

L. Huynh (), A. Nilsson (**)*

() IBM Networking Systems*

P.O. Box 12195, Research Triangle Park, NC, 27709, U.S.A.

laph@vnet.ibm.com

phone: (919) 254-4386, fax: (919) 254-4975

*(**) Electrical and Computer Engineering Department*

North Carolina State University

P.O. Box 7911, Raleigh, NC. 27695, U.S.A.

Abstract

In this paper, we present a new proportional bandwidth allocation scheme along with its analysis. The novel approach of this scheme is that it is purely end-to-end, it does not require any processing by the network, it is adaptive to changing network conditions, it has minimum oscillations, and it works even when sources have different end-to-end propagation delays. This scheme is a function of an Adaptive Rate-Based (ARB) congestion control algorithm that is designed to support best-effort traffic for which the proportional bandwidth allocation function is very desirable. We will briefly describe the ARB algorithm. We then analyze the behavior of the bottleneck queue, the settings of the ARB parameters and their effects on the behavior. This analysis will lead us to two conditions that, when they are satisfied, will provide proportional bandwidth allocation among the sources that are controlled by the ARB algorithm. We will show what parameters and how they can be controlled in order to achieve the proportional bandwidth allocation. Simulation will be used to validate our analysis.

Keywords

Adaptive rate-based congestion control, proportional bandwidth allocation, performance analysis

1.0 INTRODUCTION

Supporting best-effort traffic has become increasingly important in high-speed network. Recently, the ATM Forum has adopted an end-to-end rate-based feedback congestion control mechanism for this traffic class, also referred to as the Available Bit Rate (ABR). The reason for its importance is that this class of traffic can maximize the overall network efficiency by fully utilizing the available bandwidth in the network. One characteristic that makes this class of traffic desirable is that it can tolerate variations in the delay and bandwidth that the network can provide. But this tolerance also comes with a requirement in that the network must assure fairness in the amount of bandwidth allocated to all the connections of this traffic class, that share a network path or a bottleneck link in the network. With respect to this fairness, rather than allocating equal bandwidth to all connections, it is more desirable to assign the bandwidth based on the demand of each connection. If a connection that has twice the amount of traffic load compared with others, it should be assigned twice the amount of bandwidth when the network/path is fully utilized. We refer to such a scheme as a proportional bandwidth allocation scheme. Clearly, since best-effort traffic has to adapt to the availability of bandwidth in the network, some feedback signal that reflects the network conditions is necessary for the sources of this traffic class to control their respected bandwidth. This naturally leads to the need to have an efficient congestion control mechanism that can also provide proportional bandwidth allocation function. Recent proposals only include schemes that require extensive processing by the network (see Huynh, 1996 for a latest list of references) to provide this function. The algorithm we propose here will achieve this without any requirement on the network.

On the issue of congestion control, it has always been a focus of research and development since the initial days of computer networking. The objectives are to make more efficient use of network resources (e.g., buffer, bandwidth etc.), to enforce fairness among users, and to provide some guarantee in terms of service qualities (e.g., response time, throughput). A better congestion control mechanism would yield better throughput-cost ratio without sacrificing other objectives.

Congestion control mechanisms can basically be classified into open-loop and closed-loop schemes. Open-loop schemes require resources (i.e., bandwidth, buffers etc.) to be reserved ahead of time, whereas closed-loop implies a feedback mechanism. That is information about the status of the network is fed back to the sources of traffic, which will take actions according to these information. Clearly, the objective of these feedback mechanisms is to maximize the network efficiency by allowing sources under their control to utilize the unused bandwidth while, at the same time, minimizing loss of data. Extensive research has been done in the area of feedback congestion control mechanisms, especially recently for high-speed networks (ATM, 1996; Haas, 1991; Jacobson, 1988; Jain, 1989; Keshav, 1991; Kim, 1995; Kolarov, 1995; Mitra, 1991; Rose, 1992; to name but a few). In this paper, we describe

a new end-to-end Adaptive Rate-Based Congestion Control (ARB) mechanism that is designed based on the principles of rate control and preventive mechanism, to meet the following objectives: the algorithm is simple to implement; it is adaptive to network conditions and responsive to the user offered load; it maximizes throughput and minimizes congestion; and finally, it provides proportional bandwidth allocation function. This algorithm has been implemented in the new IBM's high performance packet routing network.

The rest of this paper is structured as follows. In section 2, we give a brief description of the algorithm. The unique feature that allows it to control the proportional bandwidth of each connection is emphasized. We then discuss the stability of the algorithm. In section 3, we present the time dependent queue analysis that takes into account the statistical variations of the arrival process, to study the behavior of the bottleneck queue and the effect of the ARB parameters on it. Based on this analysis, we propose two conditions that, when they are satisfied, will provide proportional bandwidth allocation among the connections that are controlled by the ARB algorithm. In section 4, we show the worst case throughput allocation when one or both of the conditions are not satisfied. In section 5, we present the results of our numerical and simulation study. In section 6, we discuss some future research topics.

2.0 THE ARB ALGORITHM

The ARB algorithm employs a closed-loop distributed control mechanism based on periodic exchange of information between two end-points of a full-duplex connection (with respect to ATM, two virtual channels make up a full duplex connection between two end-points). Each connection end-point, therefore, consists of two components, a sender and a receiver that cooperate to manage the connection. Periodically, the sender sends a sampling cell/packet that contains the elapsed time since its previous sampling cell (note that the sampling cell can be piggy-backed with data and not a stand-alone cell). The receiver based on its sampling cell derives the current delay along the path in the network and together with the delay information it maintains for the path, it derives an action (NORMAL, RESTRAINT, SLOWDOWN1, or SLOWDOWN2) to be taken by the sender. The actions are derived based on three thresholds T_1 , T_2 , and T_3 that are assigned to a connection based on the bottleneck link along the path of that connection. The normal operation of an ARB connection involves mainly three actions, NORMAL, RESTRAINT, and SLOWDOWN1 respectively (e.g., excessive delay, out of normal operating range) condition and to force ARB connections to react quickly. This overload condition can be caused when new connections are started in an already saturated network (or on a path), or sudden bursts of higher priority traffic (e.g., delay sensitive traffic) etc.. The sender, based on the information returned by the receiver, regulates the input traffic load accordingly. We use the phase, 'sampling cell/packet', to show that the ARB algorithm can be employed either at a lower layer (e.g., ATM adaptation layer), or

at the transport layer (e.g., TCP). In fact, a version of ARB has been implemented in a new transport protocol at IBM.

We define the following parameters in order to describe the sampling algorithm. Let Ms_n be the length of the n th sampling interval at the sender, let Mr_n be the length of the corresponding interval observed at the receiver, and let d_n be the change in network/path delay observed at the n th sampling epoch. Let D_n be the accumulated delay observed at the receiver up to the n th sampling epoch since the last rate reduction, and D_{accum} is the total accumulated delay maintained at the receiver. Figure 1(a) shows the sampling mechanism of ARB. The path delay at the n th sampling epoch is computed as follows (note that Ms_n is provided by the sender along with the sampling request): $d_n = Mr_n - Ms_n$. When $d_n > 0$, it is an indication that there is a queue build-up that is currently taking place in the network. When $d_n = 0$, it is an indication that there is no additional queue build-up since the last sampling request. And when $d_n < 0$, it is an indication that there is less queue build-up since the last sampling request. D_n is computed as: $D_n = \sum d_i$, where k represents the last epoch when D_k was reset to zero corresponding to a rate reduction at the sender (SLOWDOWN1). Please refer to (Huynh, 1996) for a detailed description of the algorithm, including the use of D_{accum} .

One important feature of ARB is that it minimizes the oscillation in the send rate (i.e., throughput) of a connection. This is accomplished by distributing the rate reduction to all connections once the queue exceeds the T_2 threshold (i.e., a source cuts its send rate by a SLOWDOWN1 factor when its sampling cell/packet detects that the queue has exceeded this threshold). Without this feature, a source may be forced to cut more times than others due to statistical behavior of the bottleneck queue with respect to the arrival of the source's sampling cells. Consequently, that source's send rate will be reduced significantly relative to others and it takes a long period for it to reclaim its share of the bandwidth. As a result, proportional bandwidth allocation can not be enforced over any reasonable amount of time.

Analysis in proving the stability of a general feedback congestion mechanism with linear increase and multiplicative decrease has recently been studied in (Altman, 1994), (Keshav, 1991), (Mukherjee, 1994). ARB belongs to this class of algorithm. Also in (Artiges, 1994), the author analytically proves the stability of the ARB algorithm based on a single source model.

The average send rate of connection i , A_i ($i=1, \dots, N$; N is the total number of connections), at steady state is given in the following equation (derived by noting that at steady state, the send rate of connection i fluctuates between the upper value and the lower with respect to the cycle of NORMAL, RESTRAINT, and SLOWDOWN1 actions respectively; see Huynh, 1996):

$$A_i = \Delta_i \frac{P_i(I)}{P_i(D)} \frac{1}{\theta_1} \left[\frac{2 - \theta_1}{2} \right]. \quad (1)$$

Where ΔI is the rate increment amount assigned to connection i . θ_1 is the rate reduction factor corresponding to SLOWDOWN1. $\frac{P_i(I)}{P_i(D)}$ is the ratio of the probability of a sampling cell that results in an increase over the probability of a sampling cell that results in a decrease, both of connection i . By keeping θ_1 and $\frac{P_i(I)}{P_i(D)}$ the same for all connections, ΔI_i becomes a simple parameter that can be manipulated to assign different proportional throughputs for different connections. The same θ_1 can easily be assigned to all the connections, whereas the $\frac{P_i(I)}{P_i(D)}$ ratio depends on other parameters of the algorithm. That is the topic of our analysis to follow.

3.0 ANALYSIS

One important objective of the analysis is to study the behaviors at the bottleneck queue, that enable the ARB algorithm to provide the function of proportional bandwidth allocation. We only provide a summarized version here, please refer to (Huynh, 1996) for details.

First, we define the following assumptions for the model: 1) cells are served at the bottleneck queue under FCFS service discipline; 2) sampling cells from all sources arrive uniformly at the bottleneck queue; 3) all sources behave according to the same ARB algorithm but can have different parameters; 4) all sources always have traffic to send. The last assumption is the situation when proportional bandwidth allocation is desirable (sources are allocated different throughput and use all of what they're allowed). It is also the traffic pattern that is commonly agreed to be a real test for a congestion control algorithm.

We will begin our analysis of the bottleneck queue with the diffusion approximation model (Newell, 1982). Figure 1(b) shows the simplified queueing model for our analysis.

Let $Q(t)$ be the bottleneck queue length at time t , and $F(q, t) = P\{Q(t) \leq q\}$ be the distribution of the queue at time t . Based on the diffusion approximation, $F(q, t)$ satisfies the following equation:

$$\frac{\partial F(q, t)}{\partial t} = -a(t) \frac{\partial F(q, t)}{\partial q} + \frac{b(t)}{2} \frac{\partial^2 F(q, t)}{\partial q^2}. \quad (2)$$

Where $a(t)$ is the instantaneous rate of change in the expected queue length and $b(t)$ is the instantaneous rate of change in the variance of the queue length at the bottleneck respectively. Let $\lambda(t)$ be the total arrival rate at time t of all the connections that send data through this bottleneck queue and are sensitive to its behavior, and $\mu(t) = \mu$ be the service rate at time t that serves traffic from these connections. Then from the above, we have $a(t) = \lambda(t) - \mu$. Our emphasis here is to study the behavior of the queue and so the time is chosen relative to the

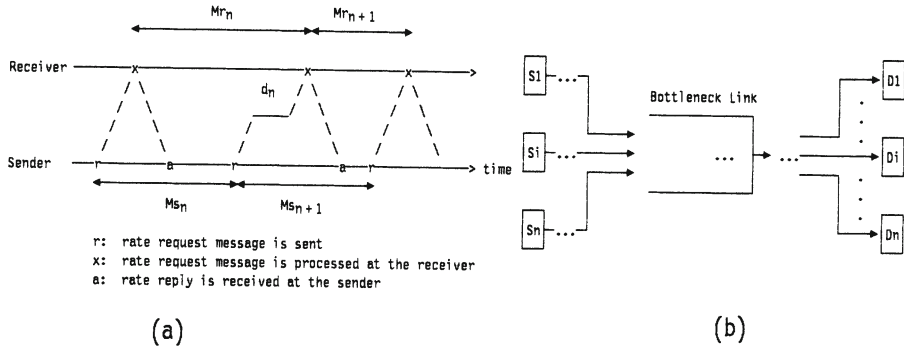


Figure 1. Rate Measurement and analysis model.

queue only. We will include propagation delay later when necessary to take into account the delayed reaction of sources.

With respect to $b(t)$, since the departure process is deterministic, we need to study the variance of the number of arrivals as a function of time (the variance of the queue length is the sum of the variance of the arrival process and the departure process). To approximate this term, we now use discrete counting process to study the variance of the arrival process. From earlier assumption, our model is such that sources always have traffic to send. Let $C(t)$ be the number of arrivals up to time t . Let s_i be the average period of source i th in this interval $(0, t]$. As a result of the arrival process, $C(t)$ will consist of a deterministic part and a random part, let's refer to the random part as $R(t)$.

Let's now consider the number of arrivals generated by a source. At a given period s_i , source i will generate $I(\frac{t}{s_i})$ cells where $I(x)$ is the integer operation of x (i.e., the floor operation). In addition, source i can generate one more cell in this time interval with the following probability:

$$z_i = \frac{t}{s_i} - I\left[\frac{t}{s_i}\right].$$

Assuming that t is uniformly distributed over source i 's period s_i . The contribution to the random number of arrivals from each source is a binomial distribution with probability z_i . The variance of this random arrival from each source is clearly $z_i(1 - z_i)$. Assuming independence between sources, the total variance due to the random arrival is:

$$\text{Var}(R(t)) = \sum_{i=1}^N z_i(1 - z_i).$$

Where N is the number of connections that directly compete for bandwidth at a bottleneck link. Clearly, this variance is bounded at $\text{Var}(R(t)) \leq B^2 = \frac{1}{4} N$ that is independent of t . As a result, we can bound the variance of the continuous time arrival process and simplify eq. (2) with $b(t) = 0$ (note that the departure process is deterministic and since

the server is always busy, its variance is zero). Therefore, the solution of eq. (2) for the queue length at the bottleneck as a function of time is thus:

$$\frac{dF(q, t | q_0, t_0)}{dq} = \frac{1}{\sqrt{2\pi} B} e^{\left[-\frac{1}{2} \left[\frac{q - q_0 - m(t, t_0)}{B} \right]^2 \right]}. \quad (3)$$

Where $m(t, t_0) = \int_{t_0}^t a(u) du$. The bottleneck queue behaves in a cyclic manner based on the behavior of the ARB sources. Each cycle contains three separate stages: the queue behavior during the rate increasing stage, the queue behavior during the rate restraining stage, and finally, the queue behavior during the rate decreasing stage. Eq. (3) allows us to study the upperbound on the stages of the queue, that are summarized in the following.

3.1 The Queue Behavior During The Increasing Stage

When the total arrival rate $\lambda(t)$ is below the bottleneck rate μ , ARB sources will increase their send rates. As $\lambda(t)$ approaches μ , the queue starts to grow. Let's anchor the initial time (i.e., $t_0 = 0$) at the point where the queue starts to grow. Let λ_0 be the total rate at t_0 and let $q_0 = 0$. The mean queue length will grow as follows:

$$m(t) = \frac{1}{2} \alpha t^2 + (\lambda_0 - \mu)t.$$

α is the rate increasing factor as described in the appendix. The queue will continue to grow until it exceeds T_1 threshold where $\lambda(t) > \mu$ and the Restraining Stage is entered. Note that when the mean queue length exceeds $T_1 + B$, it is unlikely for any source to increase since the mean queue length begins to dominate the threshold level T_1 , i.e., $F(T_1, t | q_0 = 0, t_0 = 0) \rightarrow 0$. Let P_I be the length of the increasing stage, and Q_I be the queue length at the end of this stage, then they can be approximated as follows, assuming that $\lambda_0 \simeq \mu$:

$$P_I \simeq \sqrt{\frac{2(T_1 + B)}{\alpha}} + rtt.$$

$$Q_I \simeq \frac{1}{2} \alpha \left[\sqrt{\frac{2(T_1 + B)}{\alpha}} + rtt \right]^2.$$

Where rtt is the weighted average of the roundtrip times from all the sources sharing the bottleneck link. Clearly, the send rate right before the restraining stage is entered is larger than the bottleneck service rate as the queue continues to grow. This is the maximum total arrival

rate to the bottleneck queue, λ_{\max} , it can be computed to be:
 $\lambda_{\max} = \lambda_0 + \alpha P_I$.

3.2 The Queue Behavior During The Restraining Stage

During this stage, we have $\lambda_{\max} > \mu$ and sources are restrained from increasing their rate, the corresponding bottleneck queue continues to grow until it exceeds the T_2 threshold. Again, let's anchor the relative time $t_0 = 0$ at the beginning of the restraining stage. Note that the initial queue length for this period is: $q_0 = Q_I$ from the analysis of the increasing stage. The growth in the mean queue length during this period where $t > t_0$ is simply:

$$m(t) = (\lambda_{\max} - \mu)t \quad \text{for } t > t_0.$$

When the mean queue length exceeds $T_2 + B$, it is unlikely for any source to remain restraint since the mean queue length begins to dominate the threshold level T_2 , i.e., $F(T_2, t | q_0 = Q_I, t_0 = 0) \rightarrow 0$. Let P_R be the length of the period of the restraining stage, and Q_R be the queue length at the end of this stage, they can be derived as follows:

$$P_R = \frac{(T_2 + B - Q_I, 0)^+}{\lambda_{\max} - \mu} + rtt + t_s.$$

$$Q_R = (\lambda_{\max} - \mu)P_R + Q_I.$$

Where $(x, y)^+ = \max(x, y)$, and t_s is the length of time until the arrival of the next sampling cell. The reason for this is that sources will only detect the condition of the queue exceeding $T_2 + B$ when their sampling packets arrive at the queue after the occurrence of this condition ($F(T_2, t) \rightarrow 0$). The roundtrip time is the delay until the effect of rate cutting is observed at the queue.

3.3 The Queue Behavior During The Decreasing Stage

During this stage, a source that detects the queue exceeding the T_2 threshold will cut its send rate by a factor of θ_1 . This process continues until the bottleneck queue goes below the T_2 threshold that is when sources will restraint from further cutting. The queue continues to decrease due to the fact that the send rate is less than the service rate (i.e., $\lambda(t) < \mu$). The cycle resumes when the queue goes below the T_1 threshold and sources start to increase again. The period during which the queue begins to decrease until it reaches zero can be split into two sub-periods: one is the period where the queue decreases as the sources decrease their send rates; the other is the period where the queue continues to decrease (as a result of the total send rate being less than the bottleneck service rate) toward zero but the sources have stopped decreasing. Let P_D be the total decreasing period, and P_{Dd} and

P_{Dr} be the corresponding sub-periods respectively. We will now derive the length of each sub-period.

Let's again anchor the initial time when the decreasing stage starts to be 0, the reduction in the mean queue length at time $t > t_0$ during this period is:

$$m(t) = \lambda_{\max} \left[\frac{\beta^t - 1}{\ln \beta} \right] - \mu t.$$

The initial queue for this period is $q_0 = Q_R$. The queue continues to grow beyond Q_R for as long as the total arrival rate that is on the way down is larger than the server's rate. We can compute the subperiod P_{Dd} to be:

$$P_{Dd} = (t_{Dd} + t', Ms_{\max})^- \quad \text{such that } Q_R + m(P_{Dd}) \geq 0 \\ \text{and } 0 \leq t' \leq rtt.$$

Where t_{Dd} can be computed from: $m(t_{Dd}) = (T_2 - B, 0)^+ - Q_R$. Ms_{\max} is the maximum sampling interval of all the sources. The queue length, Q_{Dd} , at the end of this P_{Dd} subperiod is: $Q_{Dd} = Q_R + m(P_{Dd})$.

Once sources restraint from further cutting, the queue continues to decrease linearly since the total arrival rate is below the server's rate. The time until the queue becomes zero is (following the restraining stage analysis):

$$P_{Dr} = \frac{-Q_{Dd}}{(\lambda_{\min} - \mu)}.$$

The total time during the decreasing stage is therefore: $P_D = P_{Dd} + P_{Dr}$.

From the above analysis, the following values are derived: RI , the rate increasing period; CT , the time for the queue to go through a complete cycle from the beginning of the increasing stage until the end of the decreasing stage; and RD , the rate decreasing period.

Let's define p_i to be the probability that upon arrival at the queue, the sampling cell will observe that the queue is under the threshold T_1 , and therefore allow the source to increase its send rate. Approximation of p_i is given in the appendix. We assume that all sources observe the same queue distribution upon the arrival of their respected sampling cells at the bottleneck queue. This assumption is reasonable considering that the bottleneck has very high utilization; the current sampling cell will observe the queue behavior that is resulted from the total traffic arrival at the bottleneck queue since the last sampling cell, including those that are from the corresponding source with the current sampling cell, minus the total traffic departed; and all sources are in the same class of users, that is they all run the same ARB algorithm.

For a given source i th with sampling interval Ms_i , its average of the number of sampling cells in the rate increasing period per queue cycle

is RI/Ms_i . The average number of sampling cells in one cycle time CT from source i is CT/Ms_i . The probability of a sampling cell that actually results in increasing the send rate (note that the source will restraint from increasing with probability of $1 - p_i$) per cycle, and therefore in general, is:

$$P_i(I) = p_i \frac{RI}{CT}.$$

In computing $P_i(D)$, the probability of a sampling cell that results in decreasing the send rate of source i , it is important to note that ARB sources will only cut once (they reset D_n after a cut) upon detection of the queue during the RD period. That is even if a source subsequent sampling cells arrive within RD after a cut, it will not reduce its send rate. The maximum total number of rate reductions during RD is therefore N , one from each source. The probability that a sampling cell from source i will result in a cut per cycle, and so in general, is therefore:

$$P_i(D) = \frac{\min(1, \frac{RD}{Ms_i})}{\frac{CT}{Ms_i}}.$$

The analysis given above yields upperbound analysis in that the standard deviation is used in all critical values of the time dependent model of ARB. It allows us to study the throughput behavior over short intervals of time where the upperbound conditions can occur (e.g., λ_{\max} and Q_R are high), and to compute the T_3 threshold. It is also useful to study the throughput ratio over a long period of time in steady state, and that requires the mean value analysis. This is done by suppressing the standard deviation in computing the ARB model equations described above. Also, from the analysis above, it is clear that there are different ARB parameters that can be used to effectively control the behavior of an ARB system as described in details in (Huynh, 1996).

The setting of T_3 threshold is for detecting when an ARB system is out of its normal operating range (e.g., abnormal conditions). For instance, when new connections are started or sudden bursts of higher priority traffic (e.g., time delay sensitive traffic) enter the network while the network is already saturated. When this condition happens, ARB will slow down significantly (SLOWDOWN2) to bring the network back to the normal operating range. As mentioned above, the upperbound analysis can be used to find an appropriate value of T_3 that has very small probability of an ARB system exceeding it during normal operation.

3.4 Conditions for Proportional Bandwidth Allocations

From the above analysis, we form two conditions that must be satisfied in order to achieve proportional bandwidth allocation. These conditions that should be evaluated according to their ordering are described in the following.

Condition 1

From eq.(1), in order to guarantee that sources will get their allocated share of the bandwidth, the ratio of $P_i(I)/P_i(D)$ must be equal, and that requires either one of the two following subconditions to be satisfied: 1) sources must have the same sampling interval; or 2) the rate decreasing period, RD , must be less than or equal to the minimum sampling interval, $RD \leq \min(Ms_1, \dots, Ms_N)$ (see Huynh, 1996).

Condition 2

In order to satisfy condition 2, either one of the following must hold: 1) sources have the same sampling interval; or 2) if sources have different sampling intervals, then $P_D + rtt \leq \min(Ms_1, \dots, Ms_N)$.

There are different ARB parameters that can be used to effectively control whether or not an ARB system satisfies the two conditions required to provide proportional bandwidth allocation. They are the increment factor ΔI_i , the decrease factor θ_i , and the T_2 threshold, and the sampling interval Ms respectively (see Huynh, 1996 for an analysis of each of them).

4.0 WORST CASE THROUGHPUT ALLOCATION

We will only provide a summary of results in this section, please refer to (Huynh, 1996) for the analysis. The following results are worst case throughput allocation when condition 1, or condition 2, or both of the proportional throughput allocation described above are not satisfied.

Let us consider two sources, i and j , respectively; and let source i have shorter sampling period than source j . In the case where condition #1 is not satisfied such that $Ms_i < RD < Ms_j$, and condition #2 is satisfied, that is $P_D + rtt \leq Ms_i$, the ratio of the throughput between source i and source j can be derived as follows:

$$\frac{A_i}{A_j} = \frac{\Delta I_i}{\Delta I_j} \frac{RD}{Ms_i}. \quad (4)$$

In the case where condition 1 is met and condition 2 is not, the throughput ratio between source i and source j is given as:

$$\frac{A_i}{A_j} = \frac{\Delta I_i}{\Delta I_j} \left[1 + \frac{1}{p_i} \frac{P_D + rtt - Ms_i}{RI} \right]. \quad (5)$$

Next, we will consider the worst case scenario where both condition #1 and condition #2 are not met. By combining the equations (4) and (5) above, we have the following worst case ratio of throughput allocation between source i and source j :

$$\frac{A_i}{A_j} = \frac{\Delta I_i}{\Delta I_j} \left[1 + \frac{1}{p_i} \frac{P_D + rtt - Ms_i}{RI} \right] \frac{RD}{Ms_i}. \quad (6)$$

5.0 NUMERICAL AND SIMULATION EXAMPLE

Due to limited space, we only show a simple configuration and discuss briefly the results. Please refer to (Huynh, 1996) for more examples and detailed analysis. In the appendix, we provide approximations of important ARB model parameters that are used to calculate our analytical results.

5.1 Case 1.a

In Figure 2, there are two campus networks, A and B respectively, connected through a WAN. There are 24 sending stations attached to the Hub/Switch A and 24 receiving stations attached to Hub B. Data are sent from stations in A to stations in B. The stations are split into two groups: group 1 consists of 8 stations (i.e., connections 1 through 8), and group 2 consists of the remaining 16 (i.e., connections 9 through 24). Connections in group 1 are allocated a throughput that is twice that of the connections in group 2.

For our first run in this configuration, the parameters are set as follows. The T_1 , T_2 , and T_3 thresholds are set at 1 cell, 20 cells, and 160 cells respectively. The unit of increment, R_{inc} , is set to 0.2% of the channel capacity, i.e., $R_{inc} = 0.002 \times 9 \text{ Mbps} = 18 \text{ Kbps}$. Group 1 connections have an increment step of $\Delta I_i = 2 \times R_{inc}$ for $i = 1, \dots, 8$. Group 2 connections have an increment step of $\Delta I_i = R_{inc}$ for $i = 9, \dots, 24$. The sampling interval is set to 30 ms for group 1 connections, and 60 ms for group 2 connections respectively. The cut factor θ_1 is set to 0.9.

Upperbound Analysis

Computation of the ARB model parameters yield the following: The resulting maximum rate is $\lambda_{\max} = 9.282 \text{ Mbps}$; the length of the rate decreasing period is $RD = 0.048 \text{ sec}$, the roundtrip time is approximately 0.01 sec; the minimum total send rate is $\lambda_{\min} = 8.45 \text{ Mbps}$; the length of the total queue decreasing stage is $P_D = 0.051 \text{ sec}$; and the length of the rate increasing period is $RI = 0.0785 \text{ sec}$. Since both conditions are not satisfied, the worst case throughput ratio between group 1 and group 2 from this upperbound analysis is, based on eq. (6), is 4.8 instead of 2 as their allocated ratio.

In using the upperbound analysis for computing the T_3 threshold, we want to obtain a threshold value such that over 99 % of the time, the

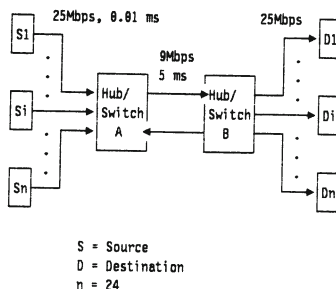


Figure 2. Proportional Bandwidth Allocation Configuration

maximum queue will be below this value. The computation yields a value of 45 cells for T_3 .

Mean Value Analysis

Based on the mean value analysis, we obtain the following values: $\lambda_{\max} = 9.2$ Mbps, $RD = 0.035$ sec, $\lambda_{\min} = 8.47$ Mbps, $P_D = 0.041$ sec, and $RI = 0.07$ sec. Again, even with the mean value analysis, the length of RD violates the condition 1 of proportional bandwidth allocation since it is larger than the sampling interval of 30 ms for the group 1 connections. Condition 2 is also violated. The worst case throughput ratio between group 1 and group 2 is therefore 3.28 instead of 2.

Simulation

Figure 3 shows the total send rate and the queue behavior at the 9 Mbps channel. Figure 4 shows the average throughput claimed by each of the connection in each group. As expected from both the upperbound and the mean value analysis, the throughput does not follow the 2 to 1 ratio that is allocated to each of the connection group. This ratio is approximately 3.2 compared with 3.28 from the mean value analysis, and 4.8 from the upperbound value analysis. Also, as expected from the upperbound analysis, no difference is observed in the final results of separate simulation runs when the upper threshold T_3 value is set to 160 cells and 45 cells respectively.

5.2 Case 1.b

For this run, we will change the ARB parameters, mainly the rate increments and the T_2 threshold, so that the proportional bandwidth allocation conditions will be satisfied. The following parameters are changed from case 1.a: the T_1 , and T_2 thresholds are set at 1 cell, and 5 cells respectively; $R_{inc} = 0.001 \times 9$ Mbps = 9 Kbps.

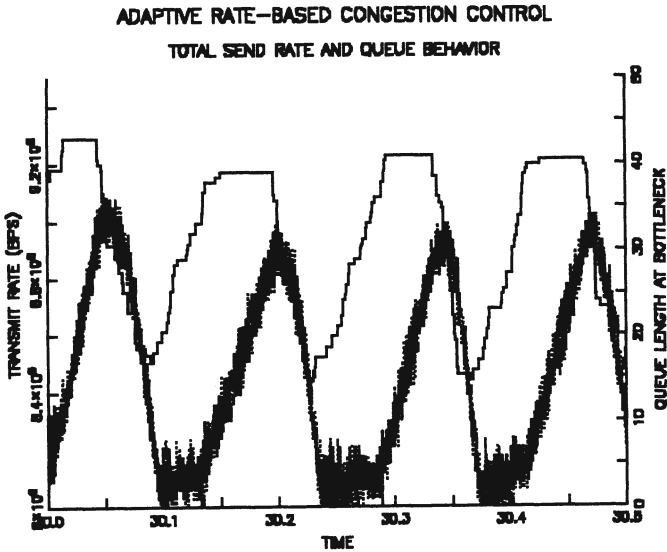


Figure 3. Total send rate and queue behavior at the 9 Mbps channel

Upperbound Analysis

The model parameters are obtained as follows: $\lambda_{\max} = 9.177$ Mbps, $RD = 0.039$ sec, $P_D = 0.03$ sec, $\lambda_{\min} = 8.425$ Mbps, and $RI = 0.144$ sec. The upperbound analysis still shows that both conditions are not satisfied. Therefore, we may expect that over some short intervals where upperbound conditions exist, the throughput will not follow the desired ratio. The worst case ratio if the upperbound conditions do exist over a period of time is 2.85 instead of the desired ratio of 2.

As in case 1.a above, our computation yields a value of 23 cells that can be used in setting T_3 .

Mean Value Analysis

Using the mean value analysis, we obtain the following values: $\lambda_{\max} = 9.12$ Mbps, $RD = 0.0276$ sec, $P_D = 0.023$ sec, $\lambda_{\min} = 8.5$ Mbps, and $RI = 0.118$ sec. Interestingly, the mean value analysis shows that the length of RD is less than the sampling interval of group 1 connections and therefore satisfies condition #1 of the proportional bandwidth allocation. For condition #2, the sum of $P_D + rtt = 0.033$ sec is still larger than the minimum sampling interval of 0.03 sec in our configuration. This is less than ideal but is still tolerable. The worst case throughput allocation ratio is 2.07 instead of the desired 2. Therefore, we can declare that condition #2 is also satisfied.

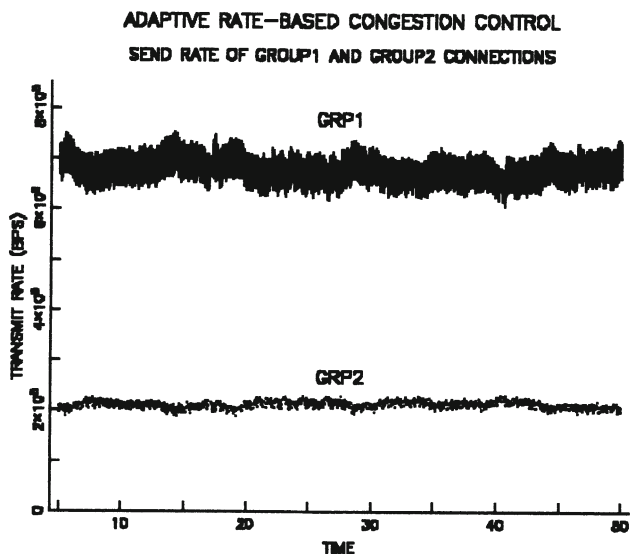


Figure 4. Average throughput for each connection in the groups (case 1.a)

Simulation

Figure 5 shows the simulation result of the average send rates of each group for case 1.b. The average throughput of each group over the simulation run follows the desired ratio of 2 to 1 (with 2 for group 1 connections). Also, as expected from the upperbound analysis, no difference is observed in the final results of separate simulation runs when the upper threshold T_3 value is set to 160 cells and 23 cells respectively.

6.0 FUTURE WORK

In high speed environment, it is desirable to take advantage of the available bandwidth as quickly as possible to minimize the time that resources are not utilized efficiently. The rate increment used in this paper is small to avoid large oscillations when the network is saturated. But the analysis suggests that the algorithm can learn when the network is not in saturation and therefore can have larger increment to take advantage of the available bandwidth in the network.

The type of traffic that we studied in this paper is homogeneous in that all connections sharing a bottleneck link are ARB type connections, that is they're all controlled by the ARB algorithm (e.g., having similar Quality of Service, QOS). Clearly, we should be able to mix other type of traffic with ARB, e.g., higher priority CBR/VBR traffic. Preliminary simulation studies show that the proportional bandwidth allocation works very well even in the presence of VBR traffic.

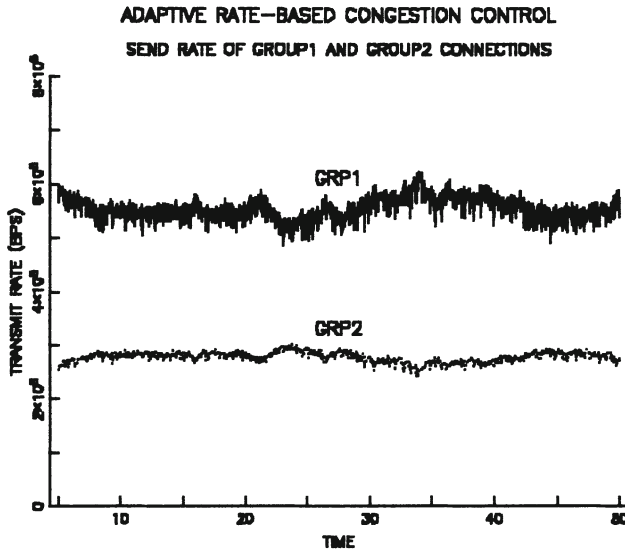


Figure 5. Average throughput for each connection in the groups (case 1.b)

7.0 APPENDIX

7.1 The Rate Increasing Factor

Let $q_{i,n}$ be the queue length seen by source i 's sampling cell at the n th epoch. Let x_n be the number of arrivals and y_n be the number of cells served during source i 's n th sampling interval, Ms_i , respectively.

The queue seen by source i 's $n+1$ st sampling cell is: $q_{i,n+1} = q_{i,n} + x_n - y_n$. An assumption is made here in that the server at the bottleneck is always busy (an assumption for the ARB model). The delay change observed by source i 's $n+1$ st sampling cell is therefore: $q_{i,n+1} - q_{i,n} = x_n - y_n$.

Clearly, the number of cells served y_n is fixed and is equal to $\frac{Ms_i}{C_t}$, where C_t is the cell transmission time. The number of arrivals, x_n , consists of two parts, a deterministic part and a random number of part. The deterministic part comes from the periodic nature of the sources, and the random part comes from the sampling interval Ms_i relative to the source j 's period s_j (i.e., s_j is just 1 over the send rate of source j during source i 's n th sampling interval. This send rate can increase and so the period s_j can change, but this change is small due to the small rate increment, it is negligible.)

The deterministic part of arrival from source j during source i 's n th sampling interval is $I\left[\frac{Ms_i}{s_j}\right]$, and the random part can be derived as shown in section 3. The total arrival, x_n , is therefore:

$$x_n = \sum_{j=1}^N I\left[\frac{Ms_i}{s_j}\right] + \sum_{j=1}^N r_j.$$

Let R_n be the random part of the total arrival x_n , then R_n has a binomial distribution. The variance of R_n has an upper bound of $\frac{1}{4}(N-1)$ (i.e., $\text{Var}(R_n) \leq \frac{1}{4}(N-1)$).

⁴ During the rate increasing period, the mean queue length is small and less than the T_1 threshold, even when the utilization is high, that is $\frac{\bar{x}_n}{\bar{y}_n} = \frac{d_n + E(R_n)}{\bar{y}_n} \simeq 1$, where d_n is the deterministic part and $E(R_n)$ is the mean of the random part of x_n respectively. Any fluctuations in the queue is mainly due to the random part of the total arrival x_n .

By using the normal distribution to approximate the binomial distribution of R_n , we can obtain the average probability of rate increase as follows:

$$\begin{aligned} p_I &\simeq P[R_n \leq E(R_n) + T_1] \\ &\simeq N\left((T_1 + \frac{1}{2})h\right). \end{aligned}$$

Where $h = 1/\sqrt{0.25(N-1)}$, and $N(x)$ is the normal distribution function (0,1). Next, we will use the approximation obtained from above for p_I to derive the increasing factor α . As described in section 2, let the rate increment of source i be ΔI_i . At each sampling epoch during the rate increasing period (R_I), source i can increase its send rate with probability p_I . As a result, source i contributes to the total rate increasing factor α by an amount of $p_I \frac{\Delta I_i}{Ms_i}$. The total contribution by all the sources are thus:

$$\alpha = p_I \sum_{i=1}^N \frac{\Delta I_i}{Ms_i}.$$

7.2 The Rate Decreasing Factor

In the following, we will derive the rate decreasing factor, β^t . Let's again split the N sources into K classes and let λ_k be the total mean arrival rate, and Ms_k be the sampling interval of sources in class k respectively, $k = 1, \dots, K$. As mentioned before, each source will cut only once during the decreasing stage so the total cut from class k is $\theta_1 \lambda_k$. This total cut distributes exponentially over the sources' sampling interval Ms_k . In other words, the rate reduction from sources in class k

is: $\theta_1 \left[\frac{t}{Ms_k} \cdot 1 \right]^- \lambda_k$, where $(x, y)^- = \min(x, y)$. The total rate reduction from all the sources is thus equal to: $\sum_{i=1}^K \theta_1 \left[\frac{t}{Ms_i} \cdot 1 \right]^- \lambda_i$.

By replacing λ_i with a portion of λ_{\max} , that is allocated to class i sources, and let f_i be this portion, then we have the following:

$$\beta^t = \sum_{i=1}^K f_i \theta_1 \left[\frac{t}{Ms_i} \cdot 1 \right]^-$$

Where $\beta_i = \theta_1 \left[\frac{t}{Ms_i} \cdot 1 \right]^-$. Although this β^t function is a sum of exponential factors of similar form (i.e., β_i^t for $i = 1, \dots, K$), it would not change the result obtained for the queue behavior during the decreasing stage in section 3. In fact, we can replace the term $\frac{\beta^t - 1}{\ln \beta}$ with the sum above:

$$\frac{\beta^t - 1}{\ln \beta} = \sum_{i=1}^K \left[\frac{\beta_i^{(t, Ms_i)} - 1}{\ln \beta_i} \right] f_i$$

8.0 REFERENCES

- Altman, E. and Baccelli, F. and Bolot, J.C. (1994) Discrete Time Analysis of Adaptive Rate Control Mechanisms. IFIP Transactions on Communications, Communication Systems (Netherlands), Volume C, No. 21.
- Artiges D. (1994) Stability Analysis of an Adaptive Rate Based Congestion Control Algorithm. NCSU Research Report.
- ATM Forum Technical Committee (1996) Traffic Management Specification Version 4.0. ATM Forum/95-0013R10 (February).
- Fratta, L. and Musumeci, L. (1995) Performance Evaluation of FR, SMDS, and ABR Services in ATM Networks. Proceedings of Sixth IFIP WG 6.3 Conference on Performance of Computer Networks (October).
- Haas, Z. (1991) Adaptive Admission Congestion Control. ACM Computer Communications Review, Volume 21, No. 5 (October).
- Huynh, L. (1996) Adaptive Rate-Based Congestion Control And Proportional Bandwidth Allocation In High-Speed Networks. Phd thesis, North Carolina State University.
- Jacobson, V. (1988) Congestion Avoidance and Control. ACM SIGCOMM'88 (August).
- Jain, R. (1989) A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks. ACM SIGCOMM'89, Volume 19, No. 5.
- Jain, R. (1992) Myths About Congestion Management in High-Speed Networks. Information Network and Data Communication IV, Elsevier Science, Publishers B.V. (North-Holland).
- Keshav, S. (1991) A Control-Theoretic Approach to Flow Control. Computer Communication Review (USA), Volume 21, No. 4 (September).

- Kim, H. and Farber, D.J. (1995) A New Congestion Control Framework for Large Bandwidth-Delay Product Networks. Proceedings of HPN VI, IFIP 6th International Conference on High Performance Network (September).
- Kolarov, A. and Ramamurthy, G. (1995) End-to-End Adaptive Rate Based Congestion Control Scheme for ABR Service in Wide Area ATM Networks. ICC'95 (June).
- Mitra, D. and Seery, J.B. (1991) Dynamic Adaptive Windows For High-Speed Data Networks With Multiple Paths and Propagation Delays (Extended Abstract). IEEE INFOCOM'91, Volume 1 (April).
- Mukherjee, A. and Strikwerda, J. (1994) Analysis of Dynamic Congestion Control Protocols -- A Fokker-Planck Approximation. Journal of High Speed Networks (Netherlands), Volume 3, No. 1.
- Newell, G.F. (1982) Applications of Queueing Theory. Second Edition, Chapman and Hall.

9.0 BIOGRAPHY

Lap T. Huynh is an Advisory Engineer in the Networking Software Division at IBM, Research Triangle Park, North Carolina. Since he started at IBM, he has worked on various networking architectures and implementations. His work and interests are in areas of congestion control for high speed networks, performance analysis, and prototyping. He obtained a BS in Electrical Engineering from University of Washington, an MS in Computer Engineering from Florida Atlantic University, and a Ph.D. in Computer Engineering at North Carolina State University in 1983, 1987, and 1996 respectively.

Arne A. Nilsson received the M.S. and Ph.D. degrees from Lund Institute of Technology, Lund, Sweden in 1968 and 1976, respectively. From 1973 to 1978 he was an Assistant Professor in the Department of Electrical Engineering at Lund Institute of Technology. In 1978 he joined the Department of Electrical and Computer Engineering at North Carolina State University, Raleigh, North Carolina, where he is now a Professor. Since 1982 he has been the Program Manager for the Networking area in the Industry/University Center for Communications and Signal Processing at NCSU. He has also been Director and Technical Director of this research Center. From 1986 to 1987 he held the position of Professor at the Royal Institute of Technology, Stockholm, Sweden. He has published more than 100 papers in technical journals and conferences and has consulted for many government agencies and corporations.

A simulation study of a wireless bandwidth reservation multiple access protocol for multimedia traffic

Zhang, Z., Habib, I. and Saadawi, T.

E.E. Department, City University of New York

New York, NY 10031, Tel. (212) 650-8378, fax (212) 650-8249

e-mail zzhang@ee-mail.engr.ccny.cuny.edu

Abstract

Bandwidth reservation multiple access protocol (BRMA) resolves access contentions for mobile users in wireless local networks. Bandwidth is assigned to each user dynamically at frame level. In each frame, request channels are deterministically assigned to each user and are therefore contention-free. Data channels are assigned dynamically by base station and are therefore collision-free. In this paper, BRMA is studied for multimedia traffic through simulations. Traffic patterns of different kinds, such as voice, data, real time high activity video (MPEG movie Star Wars) are used to study the protocol performance. With a proper admission control and with proper design of protocol parameters (service ratio, frame length, etc), traffic with different quality of service (QoS) requirements can be supported.

Keywords

Wireless networks, multiple access protocol, BRMA, multimedia

1 INTRODUCTION

To provide the wireless users with access to the advanced information services, a wireless system should be able to accommodate multimedia traffic. In a wireless system, the system structure and performance largely depend on the multiple access control (MAC) protocol that is used. So far it is still a common difficulty for a wireless system to provide the multimedia variable bit rate (VBR) traffic

with reliable service and yet maintain an efficient channel utilization. TDMA, as a mature technology in the wired systems for many years, is adapting itself to compete against the rather new CDMA to be the dominant wireless multiple access control technology in the future. However, most TDMA versions, including the multiservice dynamic reservation TDMA scheme studied in Wilson et al., (1993), do not utilize the wireless bandwidth as efficiently as the wide band CDMA especially for bursty traffic. One major reason of this is that the effect of the multiplexing gain is not maximally taken advantage of. For example, the voice activity detectors are generally not used in most TDMA systems due to the difficulty of the implementation.

Goodman et al proposed a Slotted-Aloha-reservation type packet reservation multiple access (PRMA) scheme to enhance the performance of TDMA for periodic traffic pattern such as in the case of voice conversation, see Goodman et al., (1989). After its proposal PRMA attracted much attention and was studied by many other researchers, see Goodman and Wei (1991), Nanda, (1990) and Wu et al., (1994). PRMA utilizes the speech detectors and allows a wireless user to reserve bandwidth only during talkspurts rather than during the entire period of conversation. This approach does improve the performance significantly for voice traffic. However, due to the collisions during the Slotted Aloha process before each reservation, PRMA does not perform efficiently for other types of traffic which are not periodic or which have very short bursts.

Distributed wireless random access control protocols, such as Aloha, Slotted Aloha, PRMA and its variants, work efficiently for systems with low speeds, long propagation delays and light traffic loads. They are simple to implement and provide low packet delays under light load traffic. However they cannot achieve high channel utilizations and have stability problems under heavy load. For systems with high speeds, short propagation delays and heavy loads, the centrally controlled wireless access protocols are usually better solutions at the cost of high processing.

An ATM-friendly bandwidth reservation multiple access protocol (BRMA) was proposed for wireless local high speed systems with short propagation delays, see Zhang, et al., (1996). Designed to support variable bit rate (VBR) traffic as well as other multimedia traffic, BRMA was studied through both queueing analysis and simulations for voice and data traffic in Zhang, et al., (1996). The result was that it performed better than PRMA and classical TDMA in the high speed heavy load systems.

In this paper we further study the BRMA for multimedia traffic. In section 2, we present an overview of wireless systems and the basic BRMA protocol. The applications of BRMA for two or more classes of traffic is discussed in section 3. In section 4, we study BRMA for voice and data traffic to further compare with the PRMA. In section 5, we simulate the protocol performance for voice and high activity video. Then the conclusion of this paper is given in section 6.

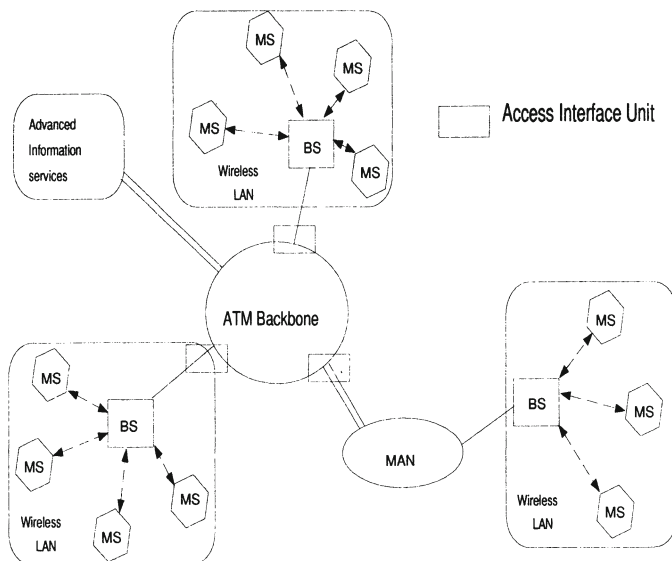


Figure 1 A system overview.

2 A WIRELESS SYSTEM OVERVIEW AND BASIC BRMA

As shown in Figure 1, wireless systems are usually connected to a wired backbone network through base stations (BS). In each wireless LAN, a number of mobile stations (MS) share the common uplink wireless channel. The downlink channel, the one from BS to MS, is an efficient broadcast channel and therefore not discussed here.

With BRMA, the uplink channel time is divided into frames of equal length. There are K data slots (data channel) and N mini slots (request channel) in each frame, where N equals to the number of MS that are connected to the BS, and where K is a design parameter. See Figure 2 for the frame structure. In each frame, each mini slot is deterministically assigned to one MS that has established a connection in the network. In each frame, each MS uses its preassigned mini slot to send its request for data slots in the current frame basing on the number of packets that are waiting in its buffer. At the end of the mini slots the BS knows the total number of requested data slots in this frame and broadcasts the data slots assignment immediately. Then each MS transmits its data packets in its assigned data slots. Apparently for this protocol to work efficiently, the packet propagation delay has to be small, such as in the case of micro cell or pico cell.

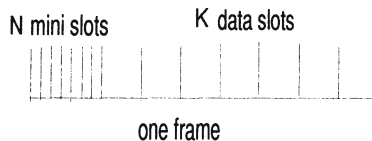


Figure 2 A BRMA frame structure.

As a central controller inside the wireless LAN, the BS keeps a connection table for all MS. When a MS is connected to or disconnected from the wireless LAN, the BS updates this information as well as the mini slots assignment instantly. In the BS, a standard ATM header will replace the short header of each data packet from the MS to the backbone network. Therefore all outgoing packets are ATM cells.

The data slot assignments are made based on the availability and request of the data slots in each frame. If the total request for data slot in this frame does not exceed K , then every MS will be assigned as many data slots as it requests. However if the total requested number exceeds K , then some MS may not get as many data slots as they need in this frame. If one MS is assigned less data slots than it requested, some of its packets will have to wait to be transmitted in the following frames. If the data packets are delay sensitive, the packet loss may occur.

To guarantee the QoS and limit the packet loss and delay, a minimum number data slots per frame is guaranteed to the MS upon request. Such minimum number is predetermined for each MS during call setup based on its traffic and QoS requirement. With such provision of minimum bandwidth, a MS will always get at least this amount of bandwidth upon request even in time of congestions. To avoid the wireless LAN being overloaded, an admission control scheme such as the one described in Habib and Saadawi (1992) should be applied.

Above data slots assignment procedure is repeated every frame and the bandwidth assignments are dynamic at the frame level. Apparently BRMA takes a good advantage of the multiplexing gain at the cost of the mini slots overhead. It suits not only the periodic traffic but also other types of multimedia traffic.

3 BRMA SUPPORTING TWO CLASSES OF TRAFFIC

Now we consider two groups (A and B) of wireless users that are connected to the BS. The number of wireless users in group A and group B are N_A and N_B respectively. Each group requires a different class of QoS. There are many definitions for QoS. Here we consider only the packet loss rate and the average packet delay. The packet loss occurs when a packet is delayed more than a limit or equivalently when the buffer overflows. For example, for voice conversations, QoS is usually met when packet loss rate is under 1% for the delay limit of 32 ms.

To guarantee the QoS to the two groups of wireless users, in each frame K_A and K_B data slots are guaranteed to group A and B respectively. Apparently we have:

$$N = N_A + N_B.$$

$$K = K_A + K_B.$$

This data slots partitioning (or called service ratio $K_A : K_B$) is determined when connections in group A and B are established, based on the traffic conditions and required QoS. When traffic conditions change, for example, when new calls are admitted or when existing calls terminate, the values of K_A and K_B need to be changed accordingly.

Due to the frame level dynamic assignment, in any particular frame, however, the data slots used by group A and B may not be exactly K_A and K_B respectively. When group A cannot use up the K_A data slots and when group B needs more than K_B data slots, group B may utilize the unused portion of the K_A data slots that are guaranteed to group A. This is true vice versa. One example of this is shown in Figure 3. Figure 4 shows another example when the service ratio $K_A : K_B$ changes.

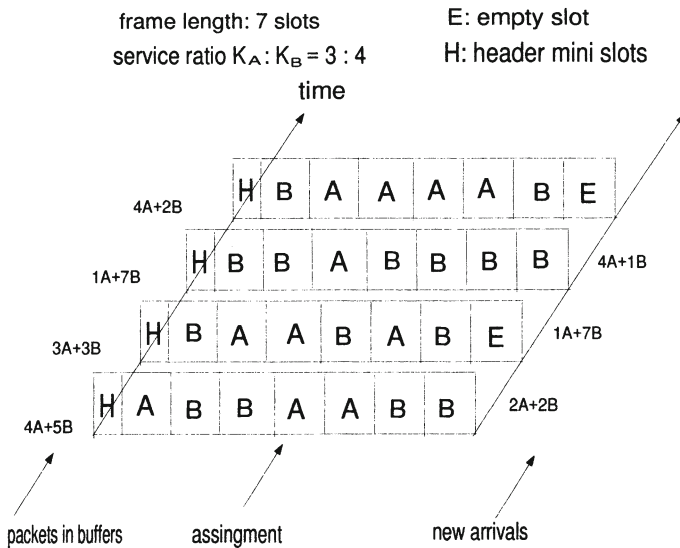


Figure 3 An example of data slots assignment.

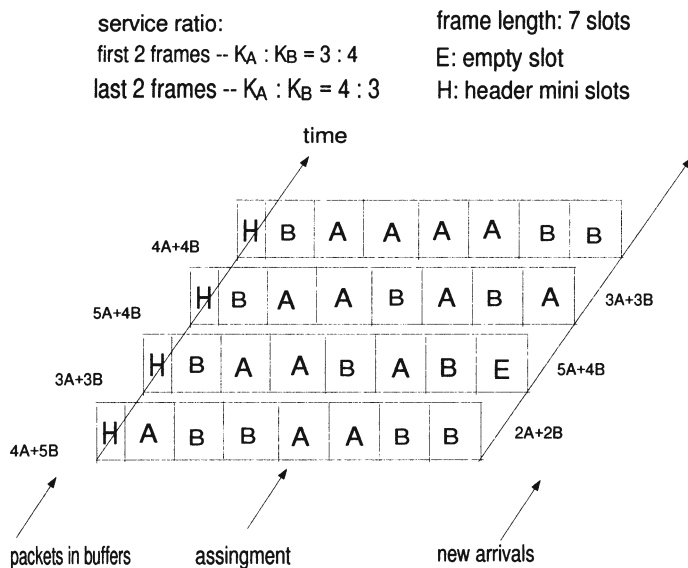


Figure 4 Another example of data slots assignment.

In the frame structure, the value of K is another design parameter. When K is too small, a large portion of the frame will be used by overhead (mini slots) and therefore the protocol efficiency is low. When K is too large, packets that arrive early in a frame will be delayed too long before data slots are requested for them therefore they will be excessively delayed. Thus an optimal value of K exists. The optimal value of K usually depends on the traffic condition and the QoS requirements, among others.

As we have seen so far, the number of classes of traffic that BRMA can support is by no means limited to two. For simplicity, we only consider the case of two classes of traffic in this paper.

4 BRMA SUPPORTING VOICE AND DATA

In this section we will do additional comparative study on BRMA for voice and Poisson data traffic, to compare with PRMA. In Wu et al., (1994), PRMA performance was further studied for voice and data traffic. A Markovian analysis was also presented in their paper. In Wu et al., (1994), voice sources are modeled by the on-off sources, and data packets are generated randomly. The inter-arrival time (length of TH mode) of the data packets is exponentially distributed with mean of 0.32 s. The data terminals generate only one single packet each time. The parameters of the system studied in Wu et al., (1994) are given in Table 1. The average delay for both voice and data packets were obtained when the packet transmission probabilities were optimized.

Variable	Nominal value
mean talkspurt duration for voice (s)	1.00
mean silent duration for voice (s)	1.35
mean duration in TH mode for data (s)	0.32
channel rate (kbps)	360
speech coding rate for voice (kbps)	32
packet size for voice and data (bits)	576
slot duration (ms)	1.6
frame duration (ms)	16
Number of slots per frame	10
priority: Pv/Pd	15
number of voice terminals Mv	5, 10, 15

Table 1 System and traffic parameters

We simulated BRMA delay performance for the same wireless system studied in Wu et al., (1994). We used the same traffic parameters as shown in Table 1, wherever applicable. In our study the service ratios are chosen as $K_V : K_D = K : 0$, which means that data packets may be transmitted only when all voice packets are transmitted and data slots in a frame are still available. Each mini slot is chosen as two bytes. Each data packet has a header of five bytes. Frame lengths are all optimized. The optimized frame length here is defined as the one that minimizes the average voice packet delay while guaranteeing that the packet loss rate for voice is under 1%.

For the traffic load shown in Figure 5, the voice packet delay performance of PRMA and BRMA are provided in Figure 6. When voice traffic load is light ($M_v=5, 10$) and when the number of data terminals is large, PRMA provides smaller average packet delay for voice. The reason of this is as following: when voice traffic load is light, packet collisions are less likely to happen at the beginning of the bursts. Once the reservation is made, following packets from the same terminal do not suffer from additional delay to the end of the burst. For BRMA, there are considerably large overhead especially due to the data terminals (each of which needs one mini slot).

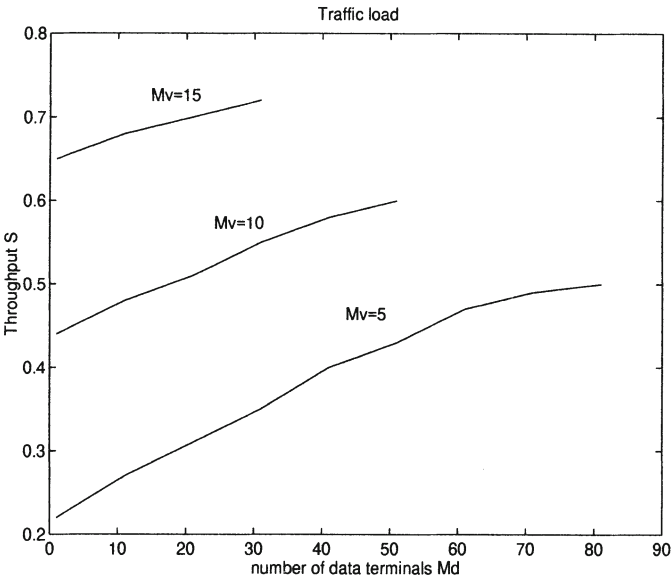


Figure 5 Traffic loads.

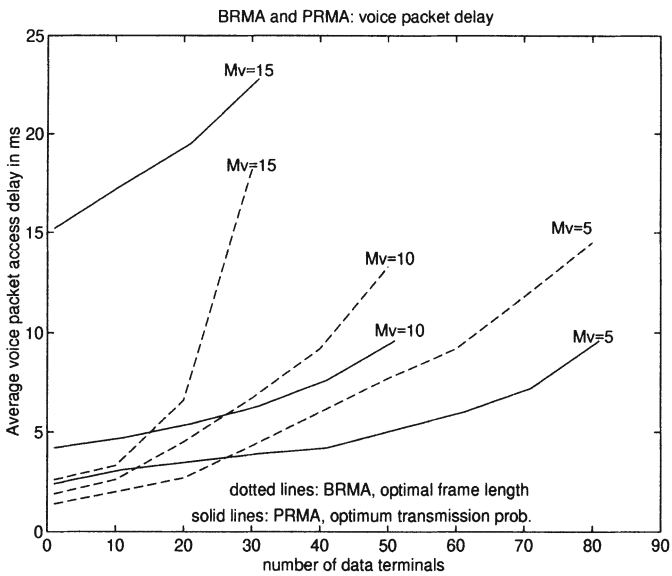


Figure 6 Voice packet delay performance for PRMA and BRMA.

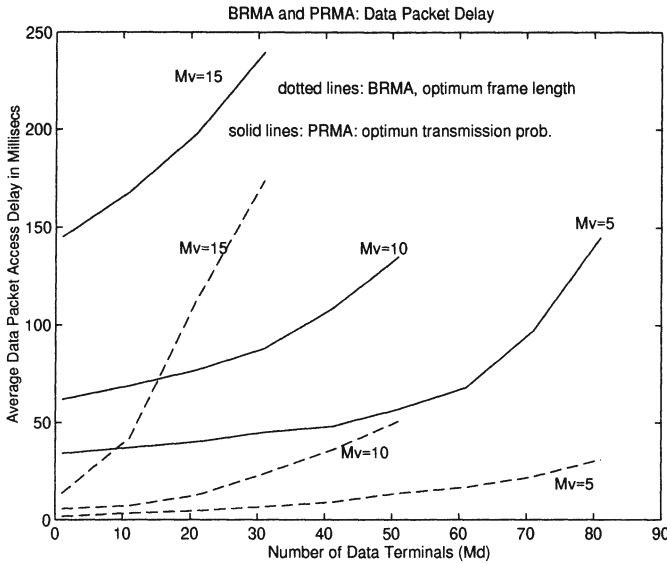


Figure 7 Data packet delay performance for PRMA and BRMA.

When the voice traffic load is higher ($M_v=15$ here), PRMA will produce higher delay than the BRMA for voice packets. This is because more voice packets collisions happen before a reservation can be made. From here we conclude that BRMA is more efficient when there are fewer wireless users in the system and each user is generating rather heavy traffic. In this case less overhead is necessary.

Figure 7 shows the data packet average delay for both PRMA and BRMA. It is clear that for $M_v = 5, 10$, and 15 , BRMA provides a better performance than PRMA for data packets. This is because that in PRMA every single data packet that is transmitted has to go through the Slotted Aloha process and no reservation can be made.

This concludes our comparative study of PRMA and BRMA: Firstly, BRMA works better for heavy load systems and PRMA works better for light load systems. This is not surprising because BRMA is a centralized control protocol while PRMA is a distributed one. Secondly, BRMA performs consistently efficiently for both periodic traffic (such as voice) and non-periodic traffic (such as data), while PRMA performs efficiently for the long-burst periodic traffic but not as efficiently for short-burst non-periodic traffic.

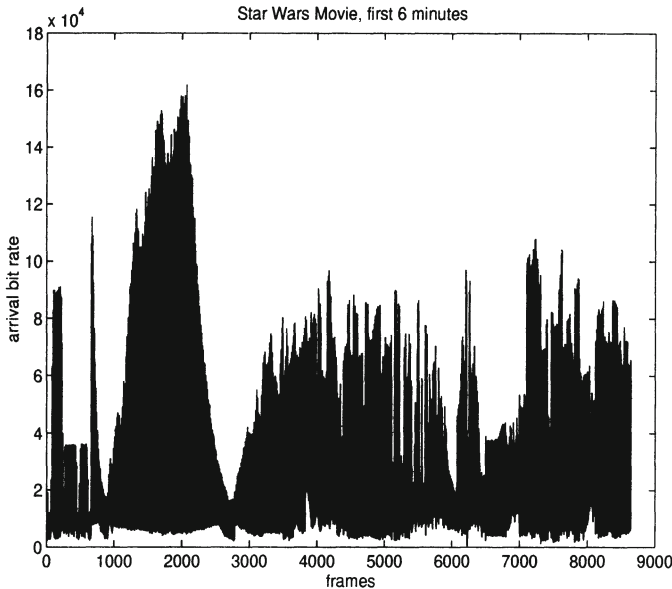


Figure 8 High activity video packet arrivals.

5 BRMA SUPPORTING HIGH ACTIVITY VIDEO AND VOICE

High activity video such as movies will be an important part of future multimedia traffic, therefore is worth investigating. It usually has high bit rate as well as high burstiness.

Currently many multimedia services in which stored media objects can be retrieved on demand by end users adopt the popular international encoding standard Motion Picture Experts Group (MPEG), see WG 11. In MPEG, video is coded into a sequence of Intracoded frame (I), Predictive frame (P) and Bidirectional frame (B). In this paper we study the BRMA performance for high activity video using the MPEG coded movie Star Wars as one example.

Our MPEG movie Star Wars is obtained from the public domain of Bellcore and the acknowledgment is due to Mark Garrett of Bellcore and Professor Martin Vetterli of the University of California at Berkeley. It is MPEG I with frame sequence of I B B P B B P B B P B B. 24 frames are generated in each second. See Garrett, (1993) for detail description of this set of data. In this study, video packets are generated uniformly in each frame interval of $1/24$ seconds.

In our study we take a part of the Star Wars movie to be a sample of high activity video source. Figure 8 shows the arrival bit rate (in bits/frame) of this source. Some statistics of this sample is shown in Table 2. We shift the sample by some frames to obtain additional video sources so that these sources are very

similar statistically. Figure 9 shows the index of dispersion for count (IDC) of the video. We see that the IDC of ten sources is much lower than that of one source, which implies that the traffic become “smoother” when multiplexed.

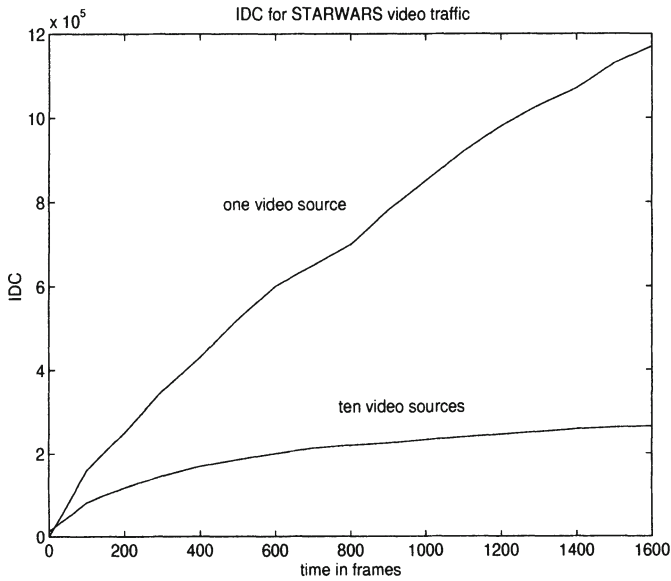


Figure 9 IDC for high activity video traffic.

peak (bit/frame)	mean (bit/frame)	variance	peak/mean
161,726	16,679	3.6e+12	9.7

Table 2 Traffic statistics for high activity video

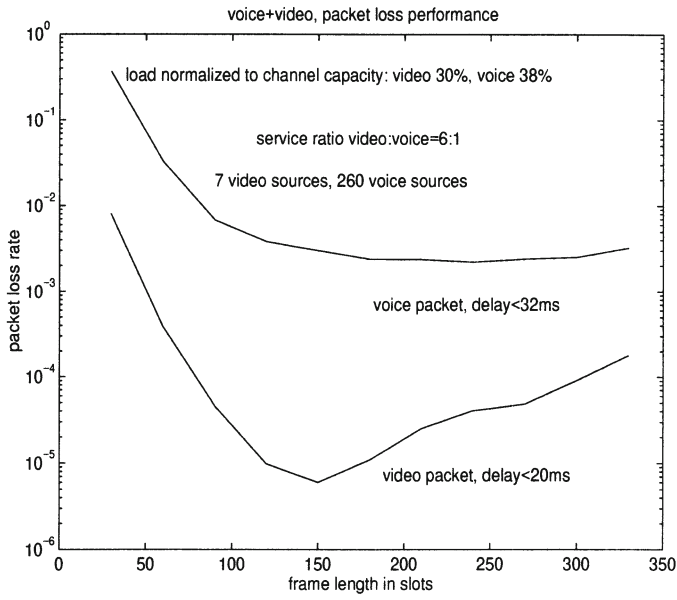


Figure 10 Packet loss performance for voice and video.

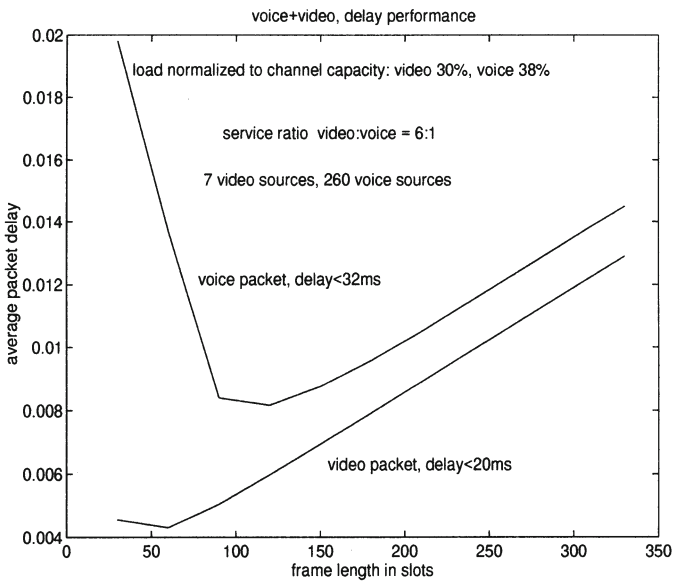


Figure 11 Packet delay performance for voice and video.

Figure 10 and 11 show the packet loss and delay performance of BRMA for high activity video and voice at the channel capacity of 10 Mbps. When the frame length is around 150 slots, 7 video and 260 voice sources can be supported with QoS that the packet loss rate for video and voice are under 1.0×10^{-5} and 1.0×10^{-2} respectively while the packet delay limits are 20 ms and 32 ms respectively. For traditional wireless TDMA, i.e., if bandwidth is dedicated to each user rather than shared, a 10 Mbps system can only support

$$N = \frac{10240000}{24 \times 161726} \approx 2.6$$

such high activity video connections. From here we see that a significant multiplexing gain is indeed achieved.

Figure 11 shows that the delays for video and voice are small when frame length is near 150 slots. Therefore the optimal frame length is near 150 slots.

Now we study the impact of service ratio on the protocol. The service ratio here sets a boundary between the resources allocated to video and voice. This boundary is flexible, as bandwidth assignment depends also on the traffic conditions. As shown in Figure 12, for the connected 7 video and 310 voice sources, the system with channel capacity 10 Mbps can guarantee the above QoS when the service ratio *video : voice* is between the values of 7.5 and 10.

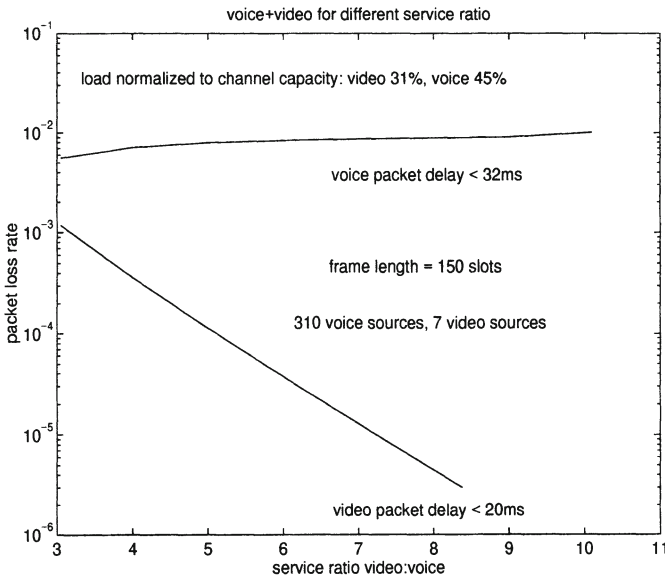


Figure 12 Packet loss performance for different service ratio.

6 CONCLUSIONS

In this paper we studied the Bandwidth Reservation Multiple Access (BRMA) protocol performance for multimedia wireless systems. We demonstrated that the protocol can support multiple classes of traffic including data, voice, and high activity video (such as MPEG movie of Star Wars). Proper frame length and service ratio guarantees that the protocol works efficiently. Admission control and minimum bandwidth that are reserved at call setup stage guarantee the QoS to each MS. It is also demonstrated in the paper that BRMA can achieve a high multiplexing gain for all types of traffic that are studied.

7 REFERENCES

- Wilson, N., Ganesh, R., Joseph, K. and Raychaudhuri, D. (1993) "Packet CDMA versus Dynamic TDMA for multiple access in an integrated voice/data PCN," IEEE JSAC, 11, 870–884.
- Garrett, M. (1993) "Contributions toward real-time services on packet networks," Ph.D Dissertation, Columbia University.
- Goodman, D., Valenzuela, R. Gayliard, K. and Ramamurthi, B. (1989) "Packet reservation multiple access for local wireless communications," IEEE Transaction on Communications, 37, 885–889.
- Goodman, D. and Wei, S. (1991) "Efficiency of packet reservation multiple access," IEEE Transaction on Vehicular Technology, 40, 170–176.
- Habib, I. Saadawi, T. (1992) "Dynamic bandwidth and access control of virtual paths in ATM networks," proceeding of IFIP 3rd International Conference on High Speed Networks, Belgian.
- Nanda, S. (1990) "Analysis of packet reservation multiple access: voice and data integration for wireless networks," Proceedings of GLOBECOM'90, San Diego, CA, 1984–1988.
- WG 11, "MPEG 1: coded representation of picture, audio and multimedia/hypermedia information," ISO/IEC JTC 1/SC 29, Nov. 1991
- Wu, G., Mukumoto, K. and Fukuka, A. (1994) "Analysis of an integrated voice and data transmission system using packet reservation multiple access," IEEE Transaction on Vehicular Technology, 43.
- Zhang, Z., Habib I. and Saadawi, T. (1996) "A bandwidth reservation multiple access protocol for wireless ATM local networks," MILCOM 96, October 1996, McLean, Virginia

8 BIOGRAPHY

Zhang, Zhiguang received his M.E. and B.E. degrees from Huazhong University of Science and Technology in China in 1989 and 1984, respectively. Currently he is a Ph.D. candidate and a teaching fellow in City University of New York. His research focuses on wireless communications.

Habib, Ibrahim is an assistant professor in City University of New York. He received his B.S. degree from Ain Shams University in Egypt, 1981. In 1984 he received an M.S. degree from Polytechnic University of Brooklyn. In 1991 he received his Ph.D. degree from City University of New York. Dr. Habib's research interests include ATM networks and applications of neural networks in high speed networks.

Saadawi, Tarek is currently a professor in City University of New York. He received his B.S. and M.S. degrees from Cairo University in 1973 and 1975, respectively. In 1980 he received a Ph.D. degree from University of Maryland. Dr. Saadawi's research interests are high-speed and multimedia networks.

INDEX OF CONTRIBUTORS

- Abdel-Wahab, H. 115
Afi, H. 130
Ahlgren, B. 249
Almeida, J. 57
Almeida, V. 57
Ammar, M.H. 3

Biersack, E. 18
Bhattacharjee, S. 265
Braun, T. 297

Calvert, K. 202, 265
Campbell, A. 166
Cavendish, D. 149
Chassot, C. 232

Diaz, M. 232
Du, D.H.C. 87

Emgrunt, H. 297

Favreau, J.P. 115
Fournier, M. 232

Gerla, M. 149
Gheorghiu, G. 101
Gumbrich, S. 297
Gustafsson, E. 280

Habib, I. 337
Hamdi, M. 130
Hsieh, J. 87
Huynh, L. 318

Kabore, P. 115
Karagiannis, G. 39
Karlsson, G. 280
Kim, O. 115
Kravets, R. 202
Krishnan, P. 202

Lamti, L. 130
Levi, A.F.J. 101
Lozes, A. 232

Micheel, J. 73
Mouftah, H.T. 217

Niemegeers, I.G.M.M. 39
Nilsson, A. 318
Nonnenmacher, J. 18

Saadawi, T. 337
Schwan, K. 202
Sivabalan, M. 217

Takahara, G. 217

Talpade, R.R. 3
Tiemann, J. 73
Tittel, C. 73
Toutain, F. 187
Tsang, R.P. 87

van Beijnum, B.J. 39

Weigmann, A. 18

Yates, D. 57

Zeadally, S. 101
Zegura, E. 265
Zhang, Z. 337

KEYWORD INDEX

- Active networks 265
- Active transport 166
- Adaptive
 - algorithms 166
 - applications 187
 - communication 202
 - rate-based congestion control 318
- Asynchronous Transfer Mode (ATM) 3,
87, 280, 297
 - routing 149
 - traffic contract 130
 - testing 73
- Bandwidth allocation 297
- B-ISDN 39
- BRMA 337
- Buffering strategies 87
- Center location 18
- Computer supported cooperative work
115
- Congestion control 39, 265
- Data partitioning 202
- Desktop conferencing 115
- Digital coded video 87
- Distributed systems 115
- Early packet discard 265
- End-to-end flow control 149
- Fair queueing 187
- Feedback implosion 18
- Handover 39
- High bandwidth applications 115
- Integrated Layer Processing (ILP) 249
- Intelligent networks 39
- Interoperability 115
- Interworking 39
- IP Multicast 3
- Java 115
- JPEG 87
- Leaky bucket algorithm 130
- Local retransmission 18
- MBONE 3
- MCS architectures 3
- Middleware 166
- Mobile communication 39, 166
- Monitoring 73
- MPEG 87, 265, 297
- Multicast 18, 115
- Multimedia 101, 337
 - application 232
 - communications 115
- Multiple access protocol 337
- Multi-service networks 217

- Network
 - modelling 217
 - performance 280
 - protocols 3
 - scalability 39
- Networking 101
- Operating systems 101
- OPNET 232
- Packet-to-cell parameter conversion 130
- Partial order connection 232
- Performance 101
 - analysis 39, 318
 - evaluation 232
 - modeling 249
- Proportional bandwidth allocation 318
- Protocol implementation techniques 249
- Queueing performance 280
- Sliding window protocol 202
- Switch architectures 87
- TCP/IP 101
- Traffic
 - characteristics 280
 - dispersion 280
 - shaping 130
- Transcoding 39
- Transport
 - protocol 232
 - service 232
- Troubleshooting 73
- Quality of Service (QoS) 166, 187
 - constraints 149
- Variable reliability 202
- VBR 297
- Video transmission 297
- Wireless networks 337